LOS ALAMOS
NATIONAL LABORATORY

Advanced Control of Operations in the Blast Furnace Project

Phase Two Effort Report: Predicting Chilled Hearth Conditions: A Classification Approach

Don R. Hush
Computer Research and Applications Group, CIC–3
Mail Stop B265

James W. Howse
Computational Science Methods Group, X–8
Mail Stop F645

LANL Technical Report: LA–UR–99–6189

Report Date: August 24, 2000

Abstract

A blast furnace is used to produce molten iron from iron oxides, coke, and flux. Ordinarily a blast furnace is controlled so that the molten iron temperature has some nominal operating value within some operating range (*e.g.*, approximately 1500°C ± 30°C for Ipsat Inland's No. 7 blast furnace). In spite of the control procedures, the iron temperature sometimes deviates from this operating range. When it does so for several ladles in succession, the condition is called a chilled hearth. When this situation arises, manual corrective actions must be taken to restore the iron temperature to its proper operating range. The corrective actions usually lead to a decrease in the production rate of molten iron. Furthermore, the lower iron temperatures usually indicate a reduction in iron quality because the trace element chemistry in the iron changes. The combination of these two factors means that iron-making companies seek to avoid chilled hearth conditions because they are economically expensive.

The overall aim of this project is to construct an automated system which detects the presence of conditions which could lead to a chilled hearth condition in a blast furnace. Reliably detecting the onset of these conditions before a chilled hearth occurs would allow corrective measures to be instituted that would probably prevent the chilled hearth. Since serious chilled hearth conditions are fairly rare in practice, this problem can be thought of in the context of anomaly detection. Intuitively, an anomaly detector takes measurements of a system as inputs, and produces a decision about whether the measurements are unusual and a confidence in that decision, as outputs. The approach to cold hearth detection discussed in this report is called the classification method. Roughly speaking, the classification method consists of designing a system which distinguishes between "normal" and "abnormal" measurements and then assigning the current measurements to either the "normal" or the "abnormal" category. The category to which the current measurements are assigned represents the current state of the blast furnace.

This report is produced in two parts. The first part is an executive summary that overviews chilled hearth conditions in a blast furnace, discusses the technical requirements associated with detecting these conditions, highlights our specific research approach to detecting chilled hearths, and presents the status of our research along with our current conclusions and future work objectives. The second part is a technical summary which goes into detail about the current state our research.

Acknowledgements

We should like to acknowledge financial support for this work from the Department of Energy's Office of Industrial Technology, Energy Efficiency and Renewable Energy, under Work Authorization ED/18019/AL04 and technical support from the blast furnace department at Ispat Inland Inc. This project has been managed by the Department of Energy's Office of Industrial Technologies Team Leader for Industrial Strategies in Steel, Scott Richlen, and the Process Lead for Industrial Processes in Steel, Gobind Jagtiani, and the Los Alamos National Laboratory Civilian and Industrial Technology Program Manager Melissa A. Miller.

We wish to acknowledge the technical contributions of Pinakin Chaubal, Paul Quisenberry, Don Zuke, and Mike Washo of Ispat Inland in helping us gather and interpret the large amounts of data required during this Phase Two task. Pinakin Chaubal served as the technical and project manager of both the Phase One and Phase Two project efforts for Ispat Inland Inc. Paul Quisenberry provided invaluable insights into the operation of the No. 7 blast furnace stoves and a number of useful suggestions concerning the advanced control technology implementation. Pinakin and Paul were also responsible for the post-audit of this technology. Mike Washo developed the interface software between the No. 7 blast furnace process monitoring and control computer system network and the workstation computer executing the software. He also provided the operating data required for the initial development and verification of the stove model and control algorithm to Los Alamos. Don Zuke provided invaluable insight and technical information on all the blast furnace sensors and operations, both under normal and off-normal operations.

We also acknowledge the support of Madhu Ranade, manager of iron making operating technology, and Wendell Carter section manager of furnace operations for the No. 7 blast furnace, for access to the No. 7 blast furnace to perform plant tests and incorporate the technology. Finally, we wish to recognize the cooperation and suggestions from the No. 7 blast furnace operators and operating foreman during the implementation effort.

Contents

Part I

Executive Summary

1 Problem Overview

A blast furnace is used to produce molten iron from iron oxides, coke, and flux. The blast furnace itself consists of a large cylinder open at the top and having ports at the bottom. Solids, such as iron ore and coke, are periodically loaded into the top of the furnace to supply the required raw materials and part of the fuel for the chemical reduction process. Some of the ports in the bottom are used to inject hot air which supplies both energy and some chemical components to drive the necessary chemical reactions. The remaining ports are used to periodically remove molten iron from the furnace. The ports for removing the molten iron are normally sealed, and when they are tapped and iron is being removed, the process is called a cast, and the molten iron is called hot metal. During a cast the hot metal is poured into containers called ladles, which are typically transported to the steel-making facility.

The temperature and trace-element chemistry of the hot metal are measured and recorded for each ladle. The trace elements typically monitored are manganese, sulfur, silicon, and titanium. The iron chemistry is monitored because the amounts of these elements affect the quality of the end products made from the iron. The hot metal temperature is monitored for two reasons. First, there is a thermal equilibrium between the iron and the slag in the blast furnace, and the temperature determines what proportion of the trace elements go into the slag as opposed to the iron. Second, the hot-metal temperature gives an indication of the location inside the blast furnace where the reduction is taking place. If the reduction occurs near the bottom of the furnace, the resulting reduction reaction is much more endothermic than when it occurs nearer the top of the furnace. This change is due to differences in the chemical constituents inside the blast furnace at different heights. Since the reaction which occurs at the bottom is quite endothermic, it draws heat out of the molten iron, thereby decreasing the hot-metal temperature. If reduction continues to occur near the bottom of the furnace, in a worst case scenario the iron and slag temperatures could become so low that these materials begin to solidify inside the blast furnace. If this occurs, the blast furnace must be shut down and the solid material must be removed from the refractory lining material. This process is extremely expensive both in terms of repair costs and lost production.

To avoid this problem, the blast furnace is controlled so that the hot-metal temperature has some nominal operating value within some operating range (*e.g.*, approximately $1500°C \pm 30°C$ for Ipsat Inland's No. 7 blast furnace). In spite of the control procedures, the hot-metal temperature sometimes deviates from this operating range. When it does so for several ladles in succession, the condition is called a chilled hearth. When this situation arises, manual corrective actions must be taken to restore the hot-metal temperature to its proper operating range. The corrective actions usually lead to a decrease in the production rate of hot metal. Furthermore, the lower hot-metal temperatures usually indicate a reduction in iron quality since the trace element amounts change. The combination of these two factors makes a chilled hearth a situation to be avoided because it is economically unprofitable.

2 Technical Objectives

The overall aim of this project is to construct an automated system which detects the presence of conditions which could lead to a chilled hearth condition in a blast furnace. Reliably detecting the onset of these conditions before a chilled hearth occurs would allow corrective measures to be instituted which would probably prevent the chilled hearth. Since serious chilled hearth conditions are fairly rare in practice, this problem can be thought of in the context of anomaly detection. Intuitively, an anomaly detector takes measurements of a system as inputs and produces a decision about whether the measurements are unusual, and a confidence in that decision, as outputs.

In the case of Ipsat Inland's No. 7 blast furnace, sensors measure the wall temperatures at various locations; the heat flux across various sections of the walls; the internal pressure at various locations; the composition, pressure, and quantity of gas released at the top of the furnace; and the quantities of various materials put into the furnace. In addition to hot-metal temperature and composition, there are roughly 200 separate measurements in total. A diagram of Ipsat Inland's No. 7 blast furnace and the particular measurements made on it are shown in Figure 1. If the hot-metal temperature and composition gave a clear indication of

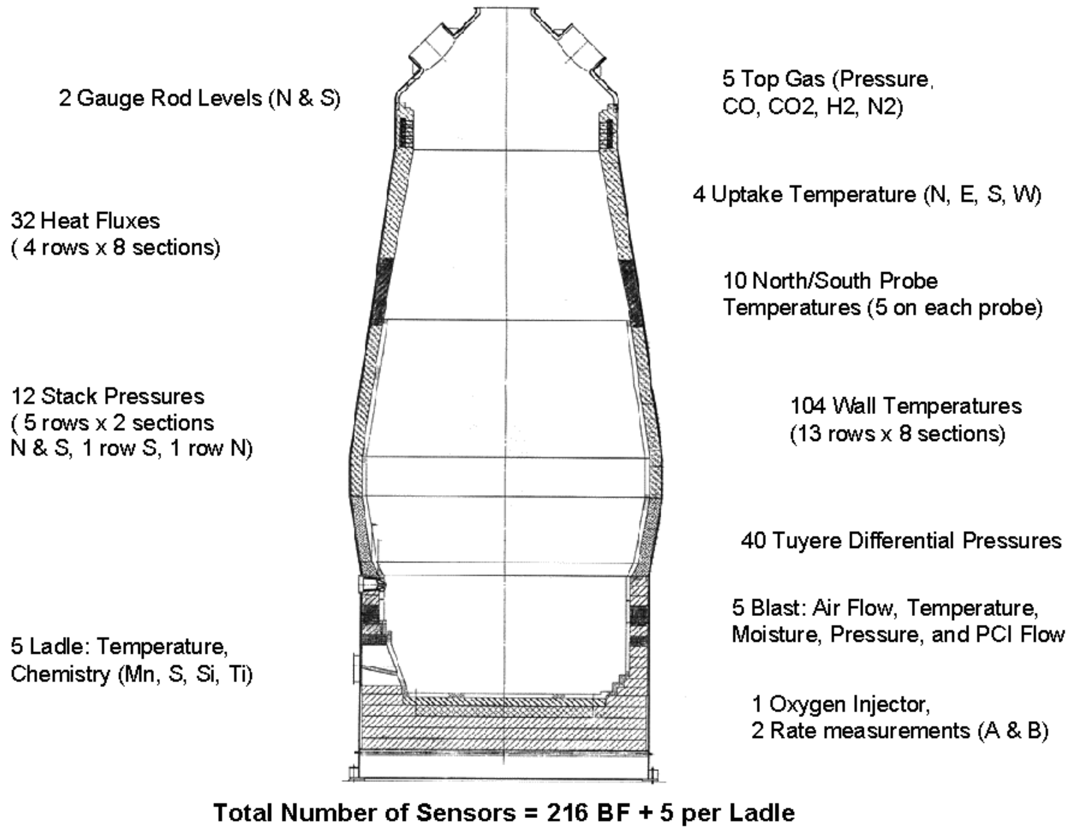
2 Gauge Rod Levels (N & S)
5 Top Gas (Pressure, CO, CO2, H2, N2)
4 Uptake Temperature (N, E, S, W)
32 Heat Fluxes (4 rows x 8 sections)
10 North/South Probe Temperatures (5 on each probe)
12 Stack Pressures (5 rows x 2 sections N & S, 1 row S, 1 row N)
104 Wall Temperatures (13 rows x 8 sections)
40 Tuyere Differential Pressures
5 Blast: Air Flow, Temperature, Moisture, Pressure, and PCI Flow
5 Ladle: Temperature, Chemistry (Mn, S, Si, Ti)
1 Oxygen Injector, 2 Rate measurements (A & B)
Total Number of Sensors = 216 BF + 5 per Ladle

Figure 1: A diagram of Ipsat Inland's No. 7 blast furnace and the associated measured quantities. This furnace is roughly 140 ft. tall and 55 ft. in diameter.

a chilled hearth *before* it occurred, then solving this problem would be fairly straightforward. Unfortunately, it appears that this set of quantities measures the effects of a chilled hearth rather than its causes, so they give no advance warning of the condition. Therefore one is forced to try to find information related to the causes of a chilled hearth in the other furnace measurements. The decision in this detection problem is whether the current state of the blast furnace is “normal” or “abnormal”. In this case “abnormal” means that the furnace is heading toward a chilled hearth, and “normal” means that it is not.

Two possible approaches to cold hearth detection are the prediction method and the classification method. Roughly speaking the prediction method consists of designing a system which predicts the hot-metal temperature based on the other furnace measurements, and then comparing this predicted temperature with the measured hot-metal temperature. If the measured and predicted temperatures differ by too much, then the current state of the blast furnace is judged to be “abnormal”; otherwise its state is “normal”. In contrast the classification method consists of designing a system which distinguishes between “normal” and “abnormal” measurements, and then assigning the current measurements to either the “normal” or the “abnormal” category. The category to which the current measurements are assigned represents the current state of the blast furnace. Only the details of the classification method are presented in this report.

3 Research Approach

The fundamental methodology underlying the classification approach to detecting chilled hearth conditions is the construction of a function which discriminates between data that precedes chilled hearth conditions, and data that does not. This approach is based on the intuition that there is some space containing functions of the measured data in which the samples that preceded chilled hearth conditions are separable from those samples that did not precede chilled hearth conditions. This approach requires the measured data to be labeled as one of two categories, that which preceded a chilled hearth is labeled “abnormal”, whereas that which does not is labeled “normal”. Naturally the dividing line between data which does and does not precede a chilled hearth is somewhat arbitrary, but some multiple (probably near unity) of the residence time of the blast furnace seems an intuitively reasonable choice. The solution to this problem can be divided into three tasks: first, deciding whether the labeled measured data is actually separable; second, constructing a subspace of measurements that preserves most of the separability; third, finding the function that separates the data in this subspace. Typically, these three tasks are called separability, feature selection, and classification respectively.

Separability can be resolved by estimating the Bayes error for the measured data, where the two categories are “normal” and “abnormal” as defined previously. The Bayes classifier is defined to be the classifier which minimizes the probability of misclassification, therefore it achieves the lowest error rate by definition. In practice, the Bayes classifier *cannot* actually be constructed, but the probability of misclassification, also called the Bayes error, that it would achieve can be estimated. For this problem, the Bayes error is the fraction of time that a “normal” data point is placed in the “abnormal” class or vice versa. If the two categories are equally likely and the Bayes error estimate is close to 50%, then this set of measurements does not contain the information needed to separate the points labeled “normal” from those labeled “abnormal”. Conversely, if the Bayes error estimate is close to 0%, then the points labeled “normal” are well separated from those labeled “abnormal”.

Feature selection can be solved by estimating the Bayes error for different subsets of the measurements, and choosing a combination that is small in size, while not leading to a large increase in the Bayes error. Since checking all possible subsets is not practical for a data set consisting of some 200 different measurements, we will use a procedure which starts with one measurement and adds measurements one at a time, provided that adding them to the subset decreases the Bayes error by a sufficiently large amount. Periodically the current subset of measurements is checked by removing measurements one at a time, if removing them from the subset increases the Bayes error by a sufficiently small amount.

Classification can be addressed by constructing a piecewise linear function which discriminates between the "normal" and "abnormal" data in the subspace chosen by the feature selection process. This classifier can then be applied on-line to continuously monitor the blast furnace for impending cold hearth conditions.

4 Research Status

The three tasks discussed in the previous section: separability, feature selection, and classification, build upon one another in a sequential fashion. It should be emphasized that there are open technical questions associated with all three of these tasks, some of which we will have to address in some fashion in order to successfully solve the chilled hearth detection problem using the classification approach. Currently, we have written a program which estimates the Bayes error from the labeled measurement data. We are currently in the final stages of debugging this program. We have not yet run this program on the blast furnace data to test how well the "normal" and "abnormal" data can be separated. Also we have not begun to address the feature selection or classification tasks, although we have existing programs that should be easily modified to solve these tasks. In short we believe that the development time will decrease as we progress through the remaining two tasks.

5 Conclusions

At present we are not certain whether chilled hearths can be detected using the classification approach, but we have encountered no insoluble problems up to this point. It should be noted that being able to accurately and efficiently estimate the Bayes error from empirical data is *extremely* useful in *any* classification problem. Therefore the Bayes error estimation program that we are currently developing will have practical utility in a wide range of application domains far beyond the current project. In order to properly conduct the separability tests, we still need additional data for 5 of the 7 examples of severe chilled hearths that were previously identified by Ipsat Inland. Specifically, we need data from Ipsat Inland for the following dates.

1. 12/7/97 12:00am - 12/11/97 12:00pm
2. 1/10/98 12:00am - 1/18/98 12:00pm
3. 1/29/98 12:00am - 2/2/98 12:00pm

Also there are some partially unresolved issues concerning the order in which ladles are taken during a cast.

6 Future Work

The following is a list of the future work that we currently anticipate, in roughly chronological order.

1. Resolve some technical issues associated with accurately and efficiently estimating the Bayes error with the current program.
2. Test the blast furnace data to determine how well the “normal” and “abnormal” data can be separated (*i.e.*, the separability task).
3. Select a subset of the blast furnace measurements which retain most of the separability between the “normal” and “abnormal” data (*i.e.*, the feature selection task).
4. Construct a system which discriminates between “normal” and “abnormal” data using this subset of measurements (*i.e.*, the classification task).
5. Install this classification based detector at Ipsat Inland and run it in real-time in an off-line mode to determine the number of false alarms and correct predictions it produces.
6. After resolving any problems associated with having too many false alarms and/or having too few correct predictions, switch over to running the system in an on-line mode.

Note that some of these tasks involve a fair number of subtasks.

Part II
Technical Summary

Notation

$\boldsymbol{x}$:	A sample vector of measurements of the state of a system.
d :	The number of measurements in the sample vector $\boldsymbol{x}$ (*i.e.*, the dimension of $\boldsymbol{x}$).
$\mathfrak{m}$:	The total number of different classes.
$\mathfrak{n}$:	The total number of sample vectors $\boldsymbol{x}$ from *all* classes (*i.e.*, $\mathfrak{n} = \sum_{i=1}^{\mathfrak{m}} \mathfrak{n}_i$).
ω_i :	The label of the ith class.
$p(\boldsymbol{x})$:	The unconditional probability density function for all sample vectors $\boldsymbol{x}$.
$p(\boldsymbol{x} \mid \omega_i)$:	The probability density function for a sample vector $\boldsymbol{x}$ conditioned on observing class ω_i.
$\mathcal{P}(\omega_i)$:	The prior probability of observing class ω_i.
$\mathcal{P}(\omega_i \mid \boldsymbol{x})$:	The posterior probability that an observed sample vector $\boldsymbol{x}$ is a member of class ω_i.
$h_i(\boldsymbol{x})$:	The discriminant function associated with class ω_i where $i = 1, \ldots, \mathfrak{m}$.
τ_i :	The classification threshold associated with class ω_i where $i = 1, \ldots, \mathfrak{m}$.
$g\big(\boldsymbol{h}(\boldsymbol{x})\big)$:	The decision function which assigns a sample $\boldsymbol{x}$ to a particular class ω_i given the discriminant function values and the threshold for $\boldsymbol{x}$.
$\mathcal{P}(\boldsymbol{x} \xrightarrow{g} \omega_j \, , \, \boldsymbol{x} \in \omega_i)$:	The probability that the sample $\boldsymbol{x}$ is a member of class ω_i and it is classified as being a member of class ω_j.
ϵ :	The average probability of misclassifying any given sample $\boldsymbol{x}$ (*i.e.*, $\epsilon = \sum_{i=1}^{\mathfrak{m}} \sum_{j \neq i} \mathcal{P}(\boldsymbol{x} \xrightarrow{g} \omega_j \, , \, \boldsymbol{x} \in \omega_i)$).
$h_i^*(\boldsymbol{x})$:	The discriminant function associated with class ω_i which produces the minimum misclassification error ϵ^*.
τ_i^* :	The classification threshold associated with class ω_i which produces the minimum misclassification error ϵ^*.
ϵ^* :	The *minimum* average probability of misclassifying any given sample $\boldsymbol{x}$. This is called the Bayes error.
$\mathcal{X}_{\omega_i}$:	A set containing the sample vectors $\boldsymbol{x}$ which are members of class ω_i.
$\mathcal{Y}_{\omega_i}$:	A set containing the labels y for the sample vectors $\boldsymbol{x}$ which are members of class ω_i.
$\mathcal{S}$:	A set containing all pairs of measurements and their respective labels (*i.e.*, $\mathcal{S} = \bigcup_{i=1}^{\mathfrak{m}} \mathcal{X}_{\omega_i} \times \bigcup_{i=1}^{\mathfrak{m}} \mathcal{Y}_{\omega_i}$). This set is called the sample set.
$\mathcal{S}_d$:	The subset of the sample set which is used to design a classifier (*i.e.*, $\mathcal{S}_d \subseteq \mathcal{S}$). This set is called the design set.
$\mathcal{S}_t$:	The subset of the sample set which is used to test a classifier (*i.e.*, $\mathcal{S}_t \subseteq \mathcal{S}$). This set is called the test set.

$h_{\mathsf{n}i}(\boldsymbol{x})$:	The discriminant function associated with class ω_i where $i = 1, \ldots, \mathsf{m}$ for a classifier designed using n samples.
$\tau_{\mathsf{n}i}$:	The classification threshold associated with class ω_i where $i = 1, \ldots, \mathsf{m}$ for a classifier designed using n samples.
ϵ_{n} :	The exact classification error for a classifier designed using n samples.
$\hat{\epsilon}_{\mathsf{n}}$:	The estimated classification error for a classifier designed using n samples and tested using a finite number of samples.
$\hat{\epsilon}_{\mathsf{L}}$:	The estimated classification error for a classifier designed and tested using the leave-one-out procedure.
$\hat{\epsilon}_{\mathsf{R}}$:	The estimated classification error for a classifier designed and tested using the resubstitution procedure.
k :	The neighborhood size used for the *kNN* classifier.
$v_k^{\omega_i}(\boldsymbol{x})$:	The volume of the neighborhood containing the k closest samples to the point $\boldsymbol{x}$ from class ω_i.
$\bar{\boldsymbol{x}}_k^{\omega_i}(\boldsymbol{x})$:	A function returning the kth nearest neighbor from class ω_i of the sample $\boldsymbol{x}$.
$\mathcal{D}_i\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_i}(\boldsymbol{x})\big)$:	The Mahalanobis distance between the point $\boldsymbol{x}$ and its kth nearest neighbor from class ω_i.
$\boldsymbol{M}_i$:	The metric matrix associated with class ω_i that is used to measure the Mahalanobis distance for that class.
$h_{kNN}(\boldsymbol{x})$:	The discriminant function associated with a *kNN* classifier for a two class problem.
τ_{kNN} :	The classification threshold associated with a *kNN* classifier for a two class problem.
ϵ_{kNN} :	The classification error associated with a *kNN* classifier for a two class problem.

1 Overview of the Classification Problem

While numerous books have been written about pattern classification, the overview presented in this section is primarily drawn from the material presented in Duda and Hart (1973), Fukunaga (1990), and Tou and Gonzalez (1974). In a classification problem, measurements of a system are assigned to a finite number of classes. For a particular set of measurements and classes, it may be impossible to achieve perfect separation. This difficulty can be illustrated with the measurements height and weight and the classes male and female. While it is generally true that men are heavier and taller than women, it does not seem reasonable to claim that all men and women can be perfectly classified on this basis alone. To make this difficulty more clear, consider only the measurement height and the class male. Clearly height will be distributed around some average value with most men lying near this average, but some lying quite far away from it. A similar situation will exist for the other three pairings of measurements and classes. This variation in the measurements will create an overlap between the height-weight distribution for men and for women. The extent of this overlap limits the degree to which men and women can be separated on the basis of these measurements. The Bayes classification error is a measure of this fundamental overlap between measurement distributions. Since this ambiguity is *irreducible* for a given set of measurements, the Bayes error is the *smallest* classification error achievable.

The intuitive argument presented above can be mathematically formalized using the notion of statistical decision theory, also called hypothesis testing or statistical inference. The number of measurements made on a particular state of the system is denoted by d, so in the previous example this dimension is $\mathsf{d} = 2$. A *sample vector*, denoted by $\boldsymbol{x} = [x_1 \; x_2 \; \cdots \; x_{\mathsf{d}}]$, is a set of measurements for a specific state belonging to a particular class, for instance the height and weight measurements for a specific man. The fact that the measurements are distributed means that $\boldsymbol{x}$ is a random variable. The number of classes is m and $\Omega = \{\omega_0, \omega_1, \ldots, \omega_{\mathsf{m}-1}\}$ is the set of all class labels. In the previous example $\mathsf{m} = 2$, and $\Omega = \{\mathsf{male}, \mathsf{female}\}$. The probability that a particular state of the system belongs to class ω_i without making any measurements, is called the *prior probability* and is denoted $\mathcal{P}(\omega_i)$. In the sense of the previous example, $\mathcal{P}(\omega_0)$ is the probability that a random person from the population is male. These probabilities reflect the frequency of occurrence of different classes in the system states. In this framework, the fact that measurements for a particular class have some distribution is described by the *conditional probability density* $p(\boldsymbol{x} \mid \omega_i)$ for all samples $\boldsymbol{x}$ from class ω_i. This is the probability of obtaining a particular set of measurements $\boldsymbol{x}$, given that the current state of the system belongs to class ω_i. In classification the class of a particular state of the system is determined from a particular sample of measurements. So the desired quantity is $\mathcal{P}(\omega_i \mid \boldsymbol{x})$, which is the probability that the state is a member of class ω_i given the sample of measurements $\boldsymbol{x}$. The relationship between this desired value and the prior probabilities $\mathcal{P}(\omega_i)$ and the conditional probability densities $p(\boldsymbol{x} \mid \omega_i)$ is

$$\mathcal{P}(\omega_i \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \omega_i)\, \mathcal{P}(\omega_i)}{\displaystyle\sum_{j=0}^{\mathsf{m}-1} p(\boldsymbol{x} \mid \omega_j)\, \mathcal{P}(\omega_j)}, \tag{1}$$

which is called *Bayes rule*. The quantity $\mathcal{P}(\omega_i \mid \boldsymbol{x})$ is called the *posterior probability*. The denominator of Equation (1) is a scaling factor to insure that $\sum_{j=0}^{\mathsf{m}-1} \mathcal{P}(\omega_j \mid \boldsymbol{x}) = 1$. The quantity $p(\boldsymbol{x})$ represents the probability of obtaining a particular set of measurements from the system, and is called the *unconditional probability density.*

In classification problems the overall goal is to assign the current state of the system to a class ω_i using the current measurement sample $\boldsymbol{x}$. One general methodology is to define a set of *discriminant functions* $h_i : \mathcal{X} \subseteq \mathbb{R}^{\mathrm{d}} \to \mathcal{H} \subseteq \mathbb{R}$ for all $i = 0, \ldots, \mathrm{m}-1$, and assign the sample vector $\boldsymbol{x}$ to class ω_i if $h_i(\boldsymbol{x}) > h_j(\boldsymbol{x})$ for all $i \neq j$. The sample vector $\boldsymbol{x}$ is assigned to a particular class by the *decision function* $g : \mathcal{H}^{\mathrm{m}} \subseteq \mathbb{R}^{\mathrm{m}} \to \{\omega_0, \ldots, \omega_{\mathrm{m}-1}\}$, which is defined as

$$g\big(\boldsymbol{h}(\boldsymbol{x})\big) = \arg \max_{i=0}^{\mathrm{m}-1}\big(h_i(\boldsymbol{x})\big). \tag{2}$$

Note that in this equation *arg* maps the element number i corresponding to the largest value of $h_i(\boldsymbol{x})$ to the appropriate class label ω_i. This slight abuse of notation is justified by the simplifications it permits in the remaining notation. Using these definitions, a classifier is an algorithm which computes m discriminant functions and selects the class corresponding to the largest discriminant function value. Conceptually the discriminant functions partition the measurement space into m regions $\mathcal{R}_0, \ldots, \mathcal{R}_{\mathrm{m}-1}$, called *decision regions.* If $g\big(\boldsymbol{h}(\boldsymbol{x})\big) = \omega_i$, then the sample $\boldsymbol{x}$ is assigned to class ω_i and $\boldsymbol{x}$ lies in region $\mathcal{R}_i$. Clearly, the choice of discriminant functions and associated decision function is *not* unique. The class assignment is not changed if all the discriminant functions are scaled by a positive constant or biased by a constant. Furthermore, if $h_i(\boldsymbol{x})$ is replaced by $f\big(h_i(\boldsymbol{x})\big)$ where $f(\cdot)$ is monotone increasing, the resulting classification is unchanged. If the decision function is also modified, then $f(\cdot)$ can be either monotone increasing or decreasing. In particular, these properties allow the number of discriminant functions to be reduced by one by either subtracting or dividing one of the discriminant functions by all of the others. In this case the decision function must be modified accordingly to match the new set of discriminant functions.

In classification, an error is made or the sample $\boldsymbol{x}$ is misclassified, if the decision function assigns $\boldsymbol{x}$ to the class ω_i when it is actually a member of some other class ω_k. The probability that the sample $\boldsymbol{x}$ is a member of class ω_k and it is classified as being a member of class $\omega_i \neq \omega_k$ is denoted $\mathcal{P}(\boldsymbol{x} \xrightarrow{g} \omega_i\,,\, \boldsymbol{x} \in \omega_k)$. The average probability of misclassification ϵ for all possible samples is

$$\epsilon = \int \sum_{k=0}^{\mathrm{m}-1} \sum_{i \neq k} \mathcal{P}(\boldsymbol{x} \xrightarrow{g} \omega_i\,,\, \boldsymbol{x} \in \omega_k)\, d\boldsymbol{x}. \tag{3}$$

The ideal classifier, called the *Bayes classifier*, is one that minimizes ϵ. It can be implemented, for example, by using the posterior probabilities as discriminant functions (*i.e.*, $h_i^*(\boldsymbol{x}) = \mathcal{P}(\omega_i \mid \boldsymbol{x})$). To see that this choice minimizes ϵ in in Equation (3), note that for a particular sample $\boldsymbol{x}$, the probability of incorrectly classifying that sample is

$$\sum_{i \neq k} \mathcal{P}(\boldsymbol{x} \xrightarrow{g^*} \omega_i\,,\, \boldsymbol{x} \in \omega_k) = \sum_{i \neq k} \mathcal{P}(\omega_i \mid \boldsymbol{x}) \qquad \text{if we assign } \boldsymbol{x} \text{ to } \omega_i \neq \omega_k. \tag{4}$$

Every time $\boldsymbol{x}$ is measured, the probability of error is minimized by choosing the class such that $g\big(\boldsymbol{h}^*(\boldsymbol{x})\big) = \omega_i$. Although the same value of $\boldsymbol{x}$ may never be measured twice, this procedure minimizes the misclassification error in Equation (3) because it minimizes it for *every* sample, hence the integral must also be minimized. The minimum misclassification error is denoted ϵ^*, and is called the *Bayes error.* These ideas are illustrated by the Bayes classifier for a 3-class problem with 1 measurement in Figure 2. The three decision regions $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}_2$ show

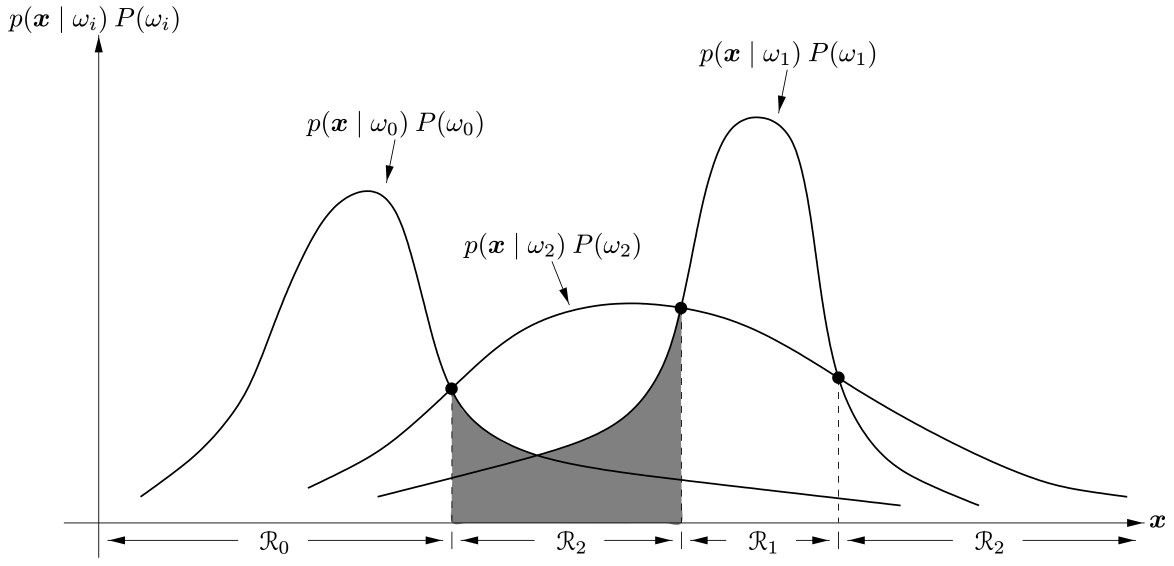

Figure 2: The decision regions for the Bayes classifier for a problem in which $\mathsf{m} = 3$ and $\mathsf{d} = 1$, when the *unscaled* posterior probabilities are as shown.

the ranges of sample values $\boldsymbol{x}$ which are assigned to classes ω_0, ω_1, and ω_2 respectively. Note that decision region $\mathcal{R}_2$ consists of two disjoint intervals. In the decision region $\mathcal{R}_i$, the value of $p(\boldsymbol{x} \mid \omega_i)\, \mathcal{P}(\omega_i)$ is always greater than these quantities for the other two classes. The dotted vertical lines show the locations of the *decision boundaries* on which $h_i^*(\boldsymbol{x}) = h_j^*(\boldsymbol{x})$, where i and j are found by noting which two curves intersect at that boundary. The curve intersections are marked with the $\bullet$ symbol. The shaded area is the probability of classifying a sample $\boldsymbol{x}$ which lies in this interval as class ω_2 and misclassifying that particular sample.

In the remainder of this paper we will only consider the special case of classification problems which have two classes labeled as $\{\omega_0, \omega_1\} = \{0, 1\}$. By making use of the previously discussed properties of discriminant functions, the two-class problem can be solved using the single discriminant function

$$\bar{h}(\boldsymbol{x}) = \frac{\mathcal{P}(\omega_1 \mid \boldsymbol{x})}{\mathcal{P}(\omega_0 \mid \boldsymbol{x})} = \frac{p(\boldsymbol{x} \mid \omega_1)}{p(\boldsymbol{x} \mid \omega_0)}\ \frac{\mathcal{P}(\omega_1)}{\mathcal{P}(\omega_0)}. \tag{5}$$

The decision rule for this discriminant function is to assign a sample $\boldsymbol{x}$ to class ω_1 when $\bar{h}(\boldsymbol{x}) > 1$ and to class ω_0 when $\bar{h}(\boldsymbol{x}) < 1$. Since the term $\frac{\mathcal{P}(\omega_0)}{\mathcal{P}(\omega_1)}$ is independent of $\boldsymbol{x}$, an alternative discriminant function and decision rule can be written as

$$h^*(\boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \omega_1)}{p(\boldsymbol{x} \mid \omega_0)} \underset{\omega_0}{\overset{\omega_1}{\gtrless}} \frac{\mathcal{P}(\omega_0)}{\mathcal{P}(\omega_1)} = \tau^*. \tag{6}$$

In this equation the operation $h^*(\boldsymbol{x}) \gtrless_{\omega_0}^{\omega_1} \tau^*$ means that $\boldsymbol{x}$ is assigned to class ω_1 when $h^*(\boldsymbol{x}) > \tau^*$ and to class ω_0 when $h^*(\boldsymbol{x}) < \tau^*$, where τ^* is called the *classifier threshold.* So the decision function associated with $h^*(\boldsymbol{x})$ is $g\big(h^*(\boldsymbol{x})\big) = \operatorname{sgn}\big(h^*(\boldsymbol{x}) - \tau^*\big)$, where through another slight abuse of notation the signum function *sgn* maps to ω_1 when its argument is positive and to ω_0 when its argument is negative. The term $\frac{p(\boldsymbol{x} \mid \omega_0)}{p(\boldsymbol{x} \mid \omega_1)}$ in Equation (6) is often called the *likelihood ratio* because $p(\boldsymbol{x} \mid \omega_i)$ represents the likelihood of class ω_i with respect to $\boldsymbol{x}$, and so the classifier defined by Equation (6) is referred to as a *likelihood ratio classifier.* Note that this likelihood ratio classifier produces exactly the same decisions and hence has the same error rate as the Bayes classifier for a two-class problem. When computing the Bayes error for a two-class problem, Equations (3) and (4) can be combined into the much simpler expression

$$\epsilon^* = \mathcal{P}(\omega_0) \int_{\mathcal{R}_1} p(\boldsymbol{x} \mid \omega_0)\, d\boldsymbol{x} + \mathcal{P}(\omega_1) \int_{\mathcal{R}_0} p(\boldsymbol{x} \mid \omega_1)\, d\boldsymbol{x}, \tag{7}$$

where each integral is only over the decision region $\mathcal{R}_i$. More useful forms of this expression for error analysis are obtained by rewriting this equation as

$$\epsilon^* = \int_{h^*(\boldsymbol{x})>\tau^*} \mathcal{P}(\omega_0 \mid \boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x} + \int_{h^*(\boldsymbol{x})<\tau^*} \mathcal{P}(\omega_1 \mid \boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x}, \tag{8}$$

$$\epsilon^* = \mathcal{P}(\omega_0) \int \mathcal{U}\big(h^*(\boldsymbol{x}) - \tau^*\big)\, p(\boldsymbol{x} \mid \omega_0)\, d\boldsymbol{x} + \mathcal{P}(\omega_1) \int \mathcal{U}\big(-[h^*(\boldsymbol{x}) - \tau^*]\big)\, p(\boldsymbol{x} \mid \omega_1)\, d\boldsymbol{x}, \tag{9}$$

$$\epsilon^* = \mathcal{P}(\omega_0)\, \mathcal{E}\big(\mathcal{I}\big(h^*(\boldsymbol{x}) > \tau^*\big) \mid \omega_0\big) + \mathcal{P}(\omega_1)\, \mathcal{E}\big(\mathcal{I}\big(h^*(\boldsymbol{x}) < \tau^*\big) \mid \omega_1\big). \tag{10}$$

In Equation (10) $\mathcal{E}(\cdot)$ is the expected value, and $\mathcal{I}(\cdot)$ is called the *indicator function*, which returns 1 when its argument is true and 0 if its argument is false. Clearly the argument of the indicator function must be a logical expression. In Equation (9) $\mathcal{U}(\cdot)$ is the *unit step function* where $\mathcal{U}(z) = \left\{\begin{smallmatrix} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{smallmatrix}\right.$. A useful interpretation of Equation (10) is that the error is the sum of the conditional means of quantities which depend only on the discriminant function and threshold. So the error of a classifier can be characterized using only the discriminant function and threshold.

2 Classification Error Overview

If the prior probabilities $\mathcal{P}(\omega_i)$ and the conditional probability densities $p(\omega_i \mid \boldsymbol{x})$ are exactly specified, then Equation (7) can be used to explicitly evaluate the Bayes error ϵ^* for the two-class problem. However even in this scenario, it may be impossible to analytically evaluate the resulting integral. In practice, the values of the prior probabilities, and the functional forms of the conditional probability densities are usually *unknown.* In this case assume that we are given two sets $\mathcal{X}_{\omega_i} = \{\boldsymbol{x}_i(1), \boldsymbol{x}_i(2), \ldots, \boldsymbol{x}_i(\mathfrak{n}_i)\} \subset \mathcal{X}$ composed of $\mathfrak{n}_i$ measurements of the system when its state was in class ω_i which are generated according to the unknown distribution $p(\boldsymbol{x} \mid \omega_i)$. Also assume that there are two sets $\mathcal{Y}_{\omega_i} = \{y_i(1), y_i(2), \ldots, y_i(\mathfrak{n}_i)\}$ composed of the the $\mathfrak{n}_i$ class labels $y_i(j) = \omega_i$ for the state of the system during each measurement. The set of all measurements is denoted $\mathcal{X}_\Omega = \mathcal{X}_{\omega_0} \cup \mathcal{X}_{\omega_1}$, and similarly the set of all labels is $\mathcal{Y}_\Omega = \mathcal{Y}_{\omega_0} \cup \mathcal{Y}_{\omega_1}$. The size of the sets $\mathcal{X}_\Omega$ and $\mathcal{Y}_\Omega$ is denoted by $\mathfrak{n} = \mathfrak{n}_0 + \mathfrak{n}_1$. The *sample set* is a combination of these two sets in which each measurement is paired with its corresponding label, which is denoted $\mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega = (\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$. So the elements of $\mathcal{S}$ are *x*-*y* pairs of the form $\big(\boldsymbol{x}_i(j), y_i(j)\big)$. We will only consider the set of discriminant functions and thresholds defined by Equation (6) such that the conditional densities $p(\boldsymbol{x} \mid \omega_i)$ and prior probabilities $\mathcal{P}(\omega_i)$ are estimated from the sample set. Furthermore we *assume* that our estimates are asymptotically consistent, so that the actual conditional densities and prior probabilities are recovered as $\mathfrak{n} \to \infty$, and therefore so are the Bayes discriminant function and threshold. Note that estimating a function, such as the conditional density $p(\boldsymbol{x} \mid \omega_i)$, in such a way that the estimate converges to the actual function as the number of samples approaches infinity is *not* as trivial as it might appear. Specifically there are many methods for which this *does not* occur, but a discussion of this subject is beyond the scope of this report. For a discussion of the topic of estimating probability densities, see the text by Silverman (1986). Let $h_{\mathfrak{n}}(\boldsymbol{x})$ and $\tau_{\mathfrak{n}}$ be the finite sample estimates of the discriminant function and threshold respectively. These $\mathfrak{n}$ sample estimates can then be substituted into Equation (9) to obtain an estimate for the Bayes error $\epsilon_{\mathfrak{n}}$, and the resulting expression for the classification error is

$$\epsilon_{\mathfrak{n}} = \mathcal{P}(\omega_0) \int \mathcal{U}\big(h_{\mathfrak{n}}(\boldsymbol{x}) - \tau_{\mathfrak{n}}\big)\, p(\boldsymbol{x} \mid \omega_0)\, d\boldsymbol{x} + \mathcal{P}(\omega_1) \int \mathcal{U}\big(-[h_{\mathfrak{n}}(\boldsymbol{x}) - \tau_{\mathfrak{n}}]\big)\, p(\boldsymbol{x} \mid \omega_1)\, d\boldsymbol{x}. \quad (11)$$

Since $h_{\mathfrak{n}}(\boldsymbol{x})$ and $\tau_{\mathfrak{n}}$ differ from $h^*(\boldsymbol{x})$ and τ^* in Equation (6) for finite $\mathfrak{n}$, this error will differ from the Bayes error ϵ^*.

Two statistics which quantify this difference are *bias* and *variance*. The bias $\mathcal{B}_{\mathfrak{n}}$ and variance $\mathcal{V}_{\mathfrak{n}}$ of the *estimate* for the classification error $\epsilon_{\mathfrak{n}}$ computed using $\mathfrak{n}$ samples are defined by

$$\mathcal{B}_{\mathfrak{n}} = \mathcal{E}_{\mathfrak{n}}\big(\epsilon_{\mathfrak{n}} - \epsilon^*\big), \quad (12)$$
$$\mathcal{V}_{\mathfrak{n}} = \mathcal{E}_{\mathfrak{n}}\big([\epsilon_{\mathfrak{n}} - \mathcal{E}_{\mathfrak{n}}(\epsilon_{\mathfrak{n}})]^2\big), \quad (13)$$

where $\mathcal{E}_{\mathfrak{n}}(\cdot)$ is the expected value over sample sets of size $\mathfrak{n}$. Intuitively, the bias is the difference between the average estimate of a quantity and the true value of that quantity. Likewise the variance is the deviation of the estimate for a quantity around the average estimate of that quantity. As $\mathfrak{n} \to \infty$ the estimates of the conditional densities and prior probabilities approach their actual values, and the classification error estimate $\epsilon_{\mathfrak{n}}$ approaches the Bayes error ϵ^*. In this case both the bias and the variance approach zero. However, since the number of measurements is finite, the density and probability estimates have nonzero biases and variances, particularly when the measurement dimension $\mathfrak{d}$ is large. Consequently, in practice, the estimate of the classification error also has a nonzero bias and variance.

Conceptually there are two different sources of error for the estimates of the Bayes error. One component is caused by using a finite number of samples to design the classifier, and the other component is the result of using a finite number of samples to test the classifier. The subset of the sample set used for classifier design $\mathcal{S}_d \subseteq \mathcal{S}$ is called the *design set*, and the subset used for design $\mathcal{S}_t \subseteq \mathcal{S}$ is the *test set*. The design set $\mathcal{S}_d$ is assumed to contain $\mathfrak{n}$ samples, and the size of the test set $\mathcal{S}_t$ is unspecified. Note that the design and test sets may be different. The expression for classification error in Equation (11) can be rewritten as

$$\epsilon_{\mathfrak{n}} = \frac{1}{2} + \frac{1}{2\,\pi} \int\!\!\int \frac{e^{j\,\nu\,(h_{\mathfrak{n}}(\boldsymbol{x}) - \tau_{\mathfrak{n}})}}{j\,\nu} \big(\mathcal{P}(\omega_0)\, p(\boldsymbol{x} \mid \omega_0) - \mathcal{P}(\omega_1)\, p(\boldsymbol{x} \mid \omega_1)\big)\, d\nu\, d\boldsymbol{x}. \quad (14)$$

This equation is obtained from Equation (11) by taking the Fourier transform, combining similar terms, and then writing the inverse Fourier transform. Let the relationship between the discriminant function and threshold produced by using a finite design sample set, and those of the Bayes classifier be $h_{\mathfrak{n}}(\boldsymbol{x}) = h^*(\boldsymbol{x}) + \Delta h(\boldsymbol{x})$ and $\tau_{\mathfrak{n}} = \tau^* + \Delta\tau$ respectively. Also for the sake of notational simplicity let $\mathsf{P}(\boldsymbol{x}) = \mathcal{P}(\omega_0)\, p(\boldsymbol{x} \mid \omega_0) - \mathcal{P}(\omega_1)\, p(\boldsymbol{x} \mid \omega_1)$. Combining the expression in Equation (9) for the Bayes error with Equation (14), the bias and variance due to a finite design set are

$$\mathcal{B}_d(\epsilon_\mathrm{n}) = \int_{h^*(\boldsymbol{x})=\tau^*} \mathcal{E}_d\big(\Delta h(\boldsymbol{x})\big)\mathsf{P}(\boldsymbol{x})\, d\boldsymbol{x} - \frac{1}{2}\frac{\partial}{\partial \xi}\left(\int_{h^*(\boldsymbol{x})=\xi} \mathcal{E}_d\big(\Delta h^2(\boldsymbol{x})\big)\mathsf{P}(\boldsymbol{x})\, d\boldsymbol{x}\right)\Bigg|_{\xi=\tau^*}, \tag{15}$$

$$\mathcal{V}_d(\epsilon_\mathrm{n}) = \int_{h^*(\boldsymbol{x})=\tau^*}\int_{h^*(\boldsymbol{y})=\tau^*} \mathcal{E}_d\big(\Delta h(\boldsymbol{x})\Delta h(\boldsymbol{y})\big)\mathsf{P}(\boldsymbol{x})\mathsf{P}(\boldsymbol{y})\, d\boldsymbol{x}\, d\boldsymbol{y}. \tag{16}$$

In this equation $\mathcal{E}_d(\cdot)$ is the expected value taken over the design sample set $\mathcal{S}_d$. These two expressions are accurate to second order and are derived using a great deal of algebra in Fukunaga (1990). Since $h^*(\boldsymbol{x}) = \tau^*$ is the Bayes decision boundary defined by Equation (6), $\mathsf{P}(\boldsymbol{x}) = 0$. This means that because the discriminant functions are chosen from Equation (6), the variance of the estimated classification error $\mathcal{V}_d(\epsilon_\mathrm{n})$ is zero *to second order.* Even in this case the bias of the estimated classification error $\mathcal{B}_d(\epsilon_\mathrm{n})$ is *not* zero, although its first term is zero. It is shown in Fukunaga (1990) that the second term in the bias expression is always positive.

If the integrals in Equation (11) could be evaluated, then an exact value for the n sample classification error ϵ_n could be obtained. However, in practice these integrals are rarely amenable to analytic evaluation. Therefore the value of ϵ_n is estimated by testing the classifier with a finite number of labeled samples using an *error-counting procedure.* In this procedure the estimate of the ω_0 error $\epsilon_{\mathrm{n}0}$ is obtained by testing each sample in $\mathcal{S}_{t0}$, counting the number of misclassified samples, and dividing by the number of samples from class ω_0, which is n_0. Similarly $\epsilon_{\mathrm{n}1}$ is estimated by testing the set $\mathcal{S}_{t1}$. This estimate for the classification error $\hat{\epsilon}_\mathrm{n}$ is

$$\hat{\epsilon}_\mathrm{n} = \frac{\mathcal{P}(\omega_0)}{\mathrm{n}_0}\sum_{j=1}^{\mathrm{n}_0}\mathcal{I}\Big(g\big(h_\mathrm{n}(\boldsymbol{x}_0(j))\big) \neq y_0(j)\Big) + \frac{\mathcal{P}(\omega_1)}{\mathrm{n}_1}\sum_{j=1}^{\mathrm{n}_1}\mathcal{I}\Big(g\big(h_\mathrm{n}(\boldsymbol{x}_1(j))\big) \neq y_1(j)\Big). \tag{17}$$

Note that the sums in Equation (17) are discrete approximations of the integrals in Equation (11). The bias and variance of the classification error estimates obtained from Equation (17) are shown by Fukunaga (1990) to be

$$\mathcal{B}_t(\hat{\epsilon}_\mathrm{n}) = \mathcal{E}_t\big(\hat{\epsilon}_\mathrm{n} - \epsilon_\mathrm{n}\big) = 0, \tag{18}$$

$$\mathcal{V}_t(\hat{\epsilon}_\mathrm{n}) = \mathcal{E}_t\big([\hat{\epsilon}_\mathrm{n} - \mathcal{E}_t(\hat{\epsilon}_\mathrm{n})]^2\big) = \frac{\big(\mathcal{P}(\omega_0)\big)^2}{\mathrm{n}_0}\epsilon_{\mathrm{n}0}(1-\epsilon_{\mathrm{n}0}) + \frac{\big(\mathcal{P}(\omega_1)\big)^2}{\mathrm{n}_1}\epsilon_{\mathrm{n}1}(1-\epsilon_{\mathrm{n}1}), \tag{19}$$

where $\mathcal{E}_t(\cdot)$ is the expected value taken over the elements in the test sample set $\mathcal{S}_t$. Note that the bias is relative to actual classification error ϵ_n, not the Bayes error ϵ^*. The estimate $\hat{\epsilon}_\mathrm{n}$ is *unbiased* because $\mathcal{E}_t(\hat{\epsilon}_\mathrm{n}) = \epsilon_\mathrm{n}$ and *consistent* since $\mathcal{V}_t(\hat{\epsilon}_\mathrm{n}) \to 0$ as $\mathrm{n}_0, \mathrm{n}_1 \to \infty$. These results imply that finite test sample size introduces variance into the classification error estimates, but by itself it introduces *no* bias.

These two sets of results from Equations (18)-(19) and Equations (15)-(16) can be combined, as in Fukunaga (1990), to analyze the bias and variance of the estimated classification error when *independent* finite size design and test sets are used. In this case, the bias $\mathcal{B}(\hat{\epsilon}_\mathrm{n})$ is entirely due to the finite number of design samples and is given by Equation (15). This means that with a finite number of samples the estimated classification error is *always* biased. The variance $\mathcal{V}(\hat{\epsilon}_{\mathsf{n}})$ is approximately equal to the sum of Equations (19) and (16). So with a finite sample size the estimated classification error *always* has a non-zero variance. Determining which effect dominates the variance can only be done if assumptions are made about the form of the classifier. For quadratic and linear classifiers, Fukunaga (1990) shows that finite test sample effects dominate the variance, but this is not a general result.

3 L and R Error Estimation Methods

In this section two methods for splitting up a sample set $\mathcal{S}$ into design and test sets are discussed. The material presented here is drawn from Fukunaga (1990) and Devroye, Györfi, and Lugosi (1996). Previously we have discussed the error estimate ϵ_{n} obtained by using a finite design sample $\mathcal{S}_d$ of size n, and the estimate of this error $\hat{\epsilon}_{\mathsf{n}}$ obtained by using a finite test sample $\mathcal{S}_t$. Our discussion assumed that all of the samples in $\mathcal{S}$ were statistically independent, and that $\mathcal{S}_d$ and $\mathcal{S}_t$ were independent. However, we have not discussed any way to actually generate $\mathcal{S}_d$ and $\mathcal{S}_t$ from a given sample set $\mathcal{S}$. The first method that we discuss for generating $\mathcal{S}_d$ and $\mathcal{S}_t$ is the *leave-one-out* (L) method, as discussed by Fukunaga (1990). This method was selected because it produces an almost unbiased estimate of ϵ_{n}. In the L method a different classifier is designed for each sample $\boldsymbol{x}(j)$ using a design set $\mathcal{S}_d(j) = \{\mathcal{S} \setminus \boldsymbol{x}(j)\}$ which excludes that sample. The classifier is then tested using the single excluded sample, therefore $\mathcal{S}_t(j) = \{\boldsymbol{x}(j)\}$. The L estimate of the classification error $\hat{\epsilon}_{\mathsf{L}}$ is obtained with the error-counting procedure discussed previously. Therefore $\hat{\epsilon}_{\mathsf{L}}$ is given by Equation (17) with $g\big(h^*(\cdot)\big) = g_{\mathsf{L}}\big(\hat{h}_{\mathsf{L}j}(\cdot)\big)$, where the subscript on $\hat{h}_{\mathsf{L}j}(\cdot)$ indicates that each sample $\boldsymbol{x}(j)$ has a different discriminant function. If all the samples in $\mathcal{S}$ are independent, then it is clear that the design and test sets for *each classifier* are independent under this method, since the sample being tested is never in the design set. One reason for using this method is expressed by the following theorem.

Theorem 1 (Devroye *et al.* (1996)). *Given that $\hat{\epsilon}_{\mathsf{L}}$ is the classification error estimate obtained by using the leave-one-out method for n sample pairs drawn from the set $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$. Then for any n*

$$\mathcal{E}_{\mathsf{n}}(\hat{\epsilon}_{\mathsf{L}}) - \epsilon_{\mathsf{n}-1} = 0.$$

This theorem means that the average value of the L error estimate is equal to the expected classification error obtained for a classifier designed with $\mathsf{n}-1$ samples. This is referred to as an *almost unbiased* estimate. While this result is very useful, it says nothing about how much a particular estimate $\hat{\epsilon}_{\mathsf{L}}$ can be expected to vary around $\epsilon_{\mathsf{n}-1}$. An asymptotic statement about this variation for a particular type of classifier is presented in the next theorem.

Theorem 2 (Devroye *et al.* (1996)). *Given that $\hat{\epsilon}_{\mathsf{L}}$ is the leave-one-out error estimate for a kNN classifier designed with n sample pairs from the set $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$. Then for every $\delta > 0$*

$$\mathcal{P}\big(|\hat{\epsilon}_{\mathsf{L}} - \epsilon_{\mathsf{n}-1}| > \delta\big) \leq 2\, e^{-\left(\frac{\mathsf{n}\,\delta^2}{k^2\,\beta^2(\mathsf{d})}\right)},$$

where $\beta(\mathsf{d})$ is a constant which depends only on the measurement dimension d. Clearly this implies that $\lim_{\mathsf{n}\to\infty} \mathcal{P}\big(|\hat{\epsilon}_{\mathsf{L}} - \epsilon_{\mathsf{n}-1}| > \delta\big) = 0$. Thus the leave-one-out error estimate for a kNN classifier is weakly consistent.

The *kNN* classifier is discussed in detail in Section 4. This result means that the variation of a particular L estimate goes to zero as the number of samples increases, when a *kNN* classifier is used. Note that a general result of this type for any form of classifier cannot be achieved. In fact, for some forms of classifier this result can be shown *not* to hold.

Because of the potentially large bias between the classification error estimate $\hat{\epsilon}_{\mathsf{L}}$ and the Bayes error ϵ^*, we would have greater confidence if we also knew upper and lower bounds for the Bayes error estimate. These bounds can be calculated by designing classifiers based on the sample set $\mathcal{S}$ using two different methods, such that one method on average is biased high and the other biased low. These two different classifiers lead to two estimates of ϵ^*, one biased high and the other low. Since the bias of $\epsilon_{\mathfrak{n}}$ in Equation (15) is always positive, $\epsilon_{\mathfrak{n}}$ itself is an upper bound for the Bayes error. This means that the L method just discussed can be used to generate an upper bound for the ϵ^*. Generating a classification error estimate that is *guaranteed* to be a lower bound of ϵ^* is extremely difficult, but it is straightforward to generate an error estimate which is guaranteed to be a lower bound of $\epsilon_{\mathfrak{n}}$. The procedure that we will use for this purpose is called the *resubstitution* (R) method, as discussed by Fukunaga (1990). In the R method all available samples are used to design the classifier $\mathcal{S}_d = \mathcal{S}$, and the same sample set is then tested $\mathcal{S}_t = \mathcal{S}$. The R estimate of the classification error is also obtained using an error-counting procedure. Therefore the R estimate for the classification error $\hat{\epsilon}_{\mathsf{R}}$ is given by Equation (17) with $g\big(h^*(\cdot)\big) = g_{\mathsf{R}}\big(\hat{h}_{\mathsf{R}}(\cdot)\big)$. It is shown in Fukunaga (1990) that $\mathcal{E}(\hat{\epsilon}_{\mathsf{R}}) \le \mathcal{E}(\hat{\epsilon}_{\mathsf{L}})$, which means that on average the R error is a lower bound for the L error. Intuitively this should be somewhat clear since in the R method each test sample is also in the design set, hence the design and test sets are dependent. On the other hand, in the L method the design and test sets are always independent, since the sample being tested is never in the design set. A more specific characterization of the difference between $\hat{\epsilon}_{\mathsf{L}}$ and $\hat{\epsilon}_{\mathsf{R}}$ for a specific type of classifier is given by the following theorem.

Theorem 3 (Devroye and Wagner (1979)). *Given that $\hat{\epsilon}_{\mathsf{L}}$ and $\hat{\epsilon}_{\mathsf{R}}$ are the leave-one-out and resubstitution error estimates, respectively, for a kNN classifier designed with $\mathfrak{n}$ sample pairs from the set $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$. Then for any $(2\,k-1) \le \mathfrak{n}-1$*

$$\mathcal{E}\big(\big[\hat{\epsilon}_{\mathsf{R}} - \epsilon_{\mathfrak{n}}\big]^2\big) - 2\,\mathcal{E}\big(\big[\hat{\epsilon}_{\mathsf{L}} - \epsilon_{\mathfrak{n}}\big]^2\big) \le \frac{8}{\sqrt{2\,\pi\,(2\,k-1)}}.$$

The *kNN* classifier is discussed in detail in Section 4. This theorem means that for sufficiently large values of k, the R error estimate $\hat{\epsilon}_{\mathsf{R}}$ becomes a reasonable estimate of both the the L error estimate $\hat{\epsilon}_{\mathsf{L}}$ and the $\mathfrak{n}$ sample classification error $\epsilon_{\mathfrak{n}}$. Unfortunately this theorem makes only an oblique statement about the relationship between $\hat{\epsilon}_{\mathsf{L}}$ and $\hat{\epsilon}_{\mathsf{R}}$. A clearer formulation of this relationship is in progress. Since we use *kNN* classifiers in computing the Bayes error estimate, both Theorems 2 and 3 are very relevant to this work.

The relationship between these two classification errors, $\hat{\epsilon}_{\mathsf{L}}$ and $\hat{\epsilon}_{\mathsf{R}}$, is shown in Figure 3 as a function of the number of samples $\mathfrak{n}$. The quantity $\mathcal{B}_{\mathfrak{n}}(\hat{\epsilon}_{\mathsf{L}} - \epsilon^*)$ is the bias of ϵ_{L} given by the sum of Equations (15) and (18). As noted in Section 2 the finite test sample does not contribute to the bias. This bias decreases as the number of samples $\mathfrak{n}$ in the design set $\mathcal{S}_d$ increases. The dark shaded band is the variance of ϵ_{L} due to the finite design set given by Equation (16). Although this variance is zero to second order, this region has been included because higher order terms are often non-negligible in the variance calculation. The light band

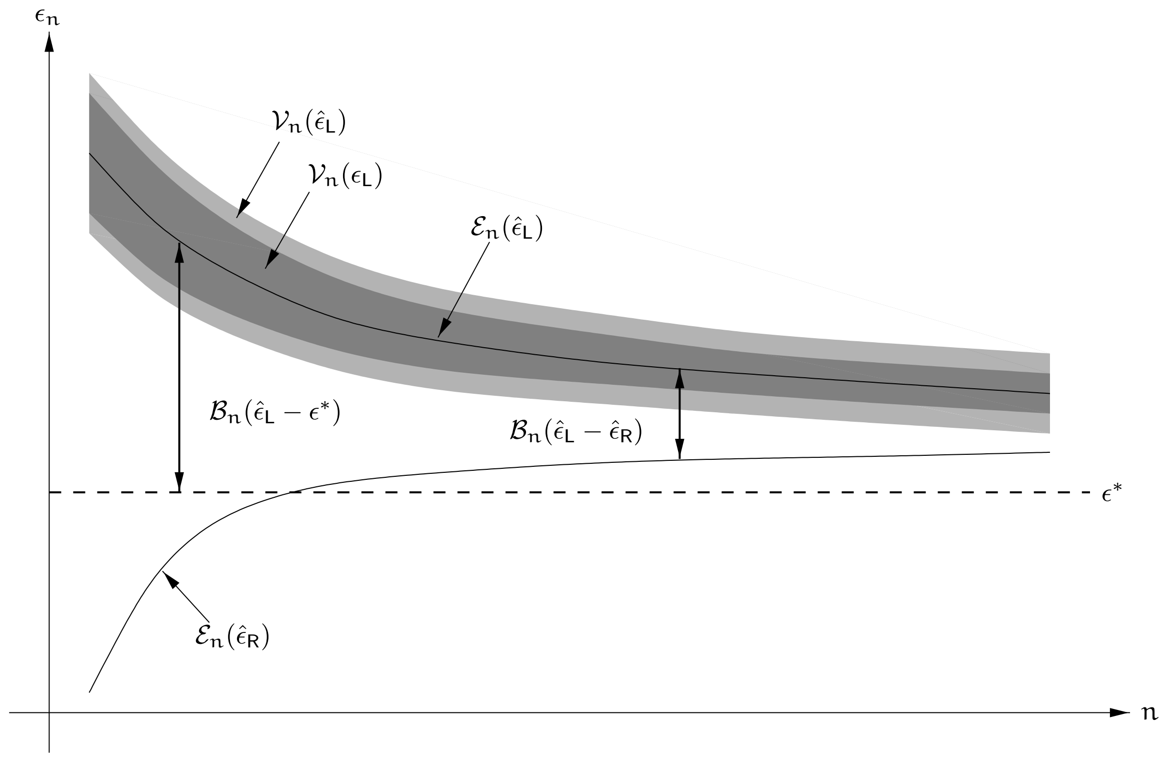
ϵ_n
$\mathcal{V}_n(\hat{\epsilon}_L)$
$\mathcal{V}_n(\epsilon_L)$
$\mathcal{E}_n(\hat{\epsilon}_L)$
$\mathcal{B}_n(\hat{\epsilon}_L - \epsilon^*)$
$\mathcal{B}_n(\hat{\epsilon}_L - \hat{\epsilon}_R)$
ϵ^*
$\mathcal{E}_n(\hat{\epsilon}_R)$
n

Figure 3: The relationship between the L error estimate $\hat{\epsilon}_L$ and the R error estimate $\hat{\epsilon}_R$. The dark shaded region is the variance associated with ϵ_L given an infinite number of test samples. The light shaded region is the additional variance incurred by using a finite number of test samples.

is the additional variance introduced by estimating $\hat{\epsilon}_L$ using a finite number of test samples given by Equation (19). The quantity $\mathcal{B}_n(\hat{\epsilon}_L - \hat{\epsilon}_R)$ is the bias between the L error and the R error. This bias also decreases as the number of samples n in the design set $\mathcal{S}_d$ increases. However, computing an explicit expression for $\mathcal{B}_n(\hat{\epsilon}_L - \hat{\epsilon}_R)$ is a non-trivial endeavor which is currently in progress. Note that if the bias $\mathcal{B}_n(\hat{\epsilon}_L - \epsilon^*)$ associated with the L error is small enough, then the R error is a lower bound for the Bayes error, but this is very difficult to guarantee.

4 k-Nearest Neighbor Classifiers

In this section the k-nearest neighbor (kNN) classifier is introduced. The material presented here is drawn from Fukunaga (1990), Devroye *et al.* (1996), and Silverman (1986). We use this type of classifier because it is shown in Devroye *et al.* (1996) that its classification error approaches the Bayes error asymptotically. Constructing the kNN classifier begins by estimating the conditional densities $\hat{p}(\boldsymbol{x} \mid \omega_i)$ in the following way. Around each sample point $\boldsymbol{x}(j)$ consider a local region with volume $v_k^{\omega_i}\big(\boldsymbol{x}(j)\big)$. The probability mass contained in this volume is approximately $\hat{p}\big(\boldsymbol{x}(j) \mid \omega_i\big)\, v_k^{\omega_i}\big(\boldsymbol{x}(j)\big)$. This probability may be approximated by considering n_i samples, computing the volume $v_k^{\omega_i}\big(\boldsymbol{x}(j)\big)$ which contains the k closest samples to the point

$\boldsymbol{x}(j)$ from class ω_i, and computing the ratio $\frac{k}{\mathfrak{n}_i}$. Therefore the density function estimate for an arbitrary sample $\boldsymbol{x}$ is

$$\hat{p}\big(\boldsymbol{x} \mid \omega_i\big) = \frac{k-1}{\mathfrak{n}_i\, v_k^{\omega_i}\big(\boldsymbol{x}\big)}, \tag{20}$$

where $k-1$ is chosen rather than k to ensure that the density estimate is unbiased, as shown in Fukunaga (1990). The nature of this density estimate around a single point $\boldsymbol{x}(j)$ is shown in Figure 4. Intuitively, at every sample the *kNN* density estimate places a window of constant

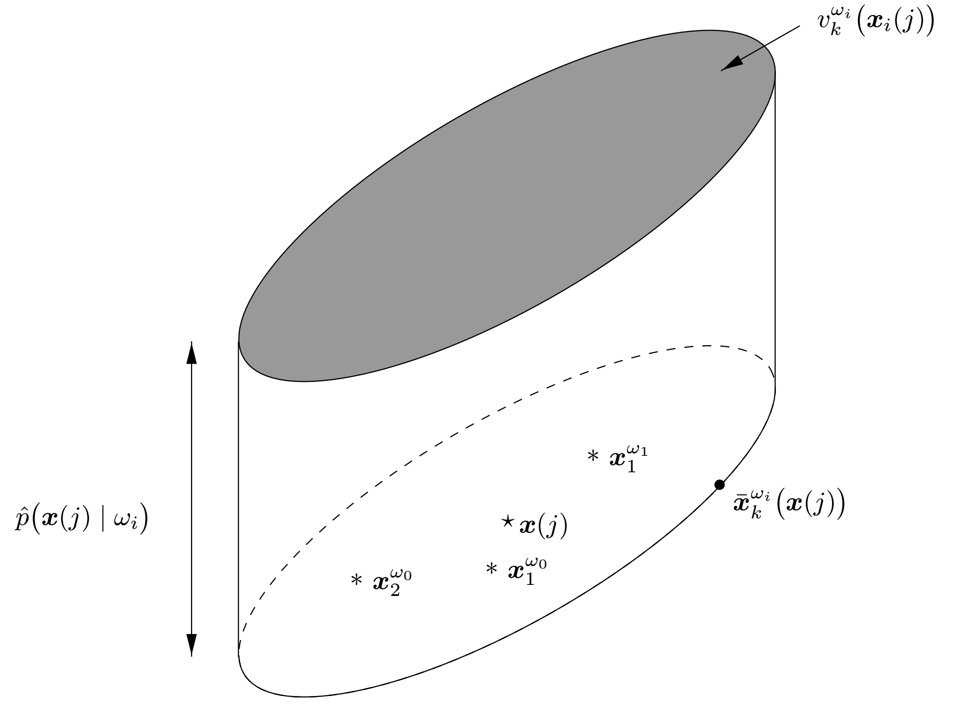
$v_k^{\omega_i}\big(\boldsymbol{x}_i(j)\big)$
$\hat{p}\big(\boldsymbol{x}(j) \mid \omega_i\big)$
$* \boldsymbol{x}_1^{\omega_1}$
$\bar{\boldsymbol{x}}_k^{\omega_i}\big(\boldsymbol{x}(j)\big)$
$\star \boldsymbol{x}(j)$
$* \boldsymbol{x}_2^{\omega_0}$
$* \boldsymbol{x}_1^{\omega_0}$

Figure 4: The *kNN* density estimate $\hat{p}\big(\boldsymbol{x}(j) \mid \omega_i\big)$ for the sample point $\boldsymbol{x}(j)$. The shaded region is the volume of the hyper-ellipsoid bounded by the kth closest point from class ω_i. The height of this window is the density estimate.

height which has a hyper-ellipsoidal footprint. The height of the window is computed from Equation (20). The size of the hyper-ellipsoid is determined by the distance from the sample point to the kth closest point from class ω_i, and therefore varies with the sample point being considered. The form of the distance measure determines the specific shape of the hyper-ellipsoid. The variable k represents the neighborhood size.

It is proved by Fukunaga (1990) that the volume for the hyper-ellipsoid in *kNN* is

$$v_k^{\omega_i}(\boldsymbol{x}) = \pi^{\frac{\mathrm{d}}{2}}\, \Gamma^{-1}\left(\frac{\mathrm{d}}{2}+1\right)\, \big|\boldsymbol{M}_i\big|^{\frac{1}{2}}\, \mathcal{D}_i^{\mathrm{d}}\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_i}(\boldsymbol{x})\big), \tag{21}$$

where $\Gamma^{-1}(\cdot)$ is the inverse of the Gamma function, and $\bar{\boldsymbol{x}}_k^{\omega_i}(\boldsymbol{x})$ is a function which returns the kth nearest neighbor from class ω_i of the input sample $\boldsymbol{x}$. The quantity $\big|\boldsymbol{M}_i\big|$ is the determinant of the *metric matrix* for class ω_i where $\boldsymbol{M}_i$ is symmetric and positive definite. One possible choice for the metric matrix is the *covariance matrix* $\boldsymbol{\Sigma}_i$ associated with class ω_i. If the actual covariance matrices are unknown they can be estimated from the samples by

$$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{\mathfrak{n}_i - 1} \sum_{j=1}^{\mathfrak{n}_i} \big(\boldsymbol{x}_i(j) - \hat{\boldsymbol{\mu}}_i\big) \big(\boldsymbol{x}_i(j) - \hat{\boldsymbol{\mu}}_i\big)^T, \tag{22}$$

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{\mathfrak{n}_i} \sum_{j=1}^{\mathfrak{n}_i} \boldsymbol{x}_i(j). \tag{23}$$

The quantity $\hat{\boldsymbol{\mu}}_i$ is the sample mean for class ω_i and the notation $\boldsymbol{w}\,\boldsymbol{z}^T$ denotes the outer product of two vectors. Note that the sample covariance matrix is symmetric and positive semi-definite. The function $\mathcal{D}_i\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_i}(\boldsymbol{x})\big)$ in Equation (21) is the *Mahalanobis distance* between the sample point $\boldsymbol{x}$ and its kth nearest neighbor from class ω_i, which is computed by

$$\mathcal{D}_i(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{M}_i\,(\boldsymbol{x} - \boldsymbol{y})}. \tag{24}$$

The notation $\boldsymbol{w}^T\boldsymbol{z}$ denotes the inner product of two vectors. It is straightforward to show that the surfaces of constant distance under the Mahalanobis metric are hyper-ellipses, which is why the window footprint of the *kNN* density estimate is hyper-ellipsoidal.

The discriminant function and decision rule for the *kNN* classifier are obtained by substituting Equations (20) and (21) into Equation (6). This leads to the expression

$$\begin{aligned}
&\frac{k-1}{\mathfrak{n}_1\, v_k^{\omega_1}(\boldsymbol{x})}\,\frac{\mathfrak{n}_0\, v_k^{\omega_0}(\boldsymbol{x})}{k-1} \underset{\omega_0}{\overset{\omega_1}{\gtrless}} \frac{\mathcal{P}(\omega_0)}{\mathcal{P}(\omega_1)}, \\
&\therefore h_{kNN}(\boldsymbol{x}) = \frac{\mathcal{D}_0^2\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_0}(\boldsymbol{x})\big)}{\mathcal{D}_1^2\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_1}(\boldsymbol{x})\big)} \underset{\omega_0}{\overset{\omega_1}{\gtrless}} \left(\frac{\mathfrak{n}_1\,|\boldsymbol{M}_1|}{\mathfrak{n}_0\,|\boldsymbol{M}_0|}\,\frac{\mathcal{P}(\omega_0)}{\mathcal{P}(\omega_1)}\right)^{\frac{2}{\mathrm{d}}} = \tau_{kNN}, \\
&\Rightarrow \left(\frac{\mathfrak{n}_0\,|\boldsymbol{M}_0|}{\mathcal{P}(\omega_0)}\right)^{\frac{1}{\mathrm{d}}} \mathcal{D}_0\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_0}(\boldsymbol{x})\big) \underset{\omega_0}{\overset{\omega_1}{\gtrless}} \left(\frac{\mathfrak{n}_1\,|\boldsymbol{M}_1|}{\mathcal{P}(\omega_1)}\right)^{\frac{1}{\mathrm{d}}} \mathcal{D}_1\big(\boldsymbol{x}, \bar{\boldsymbol{x}}_k^{\omega_1}(\boldsymbol{x})\big).
\end{aligned} \tag{25}$$

The *kNN* decision rule can be interpreted from the last line of this equation. If the terms $\left(\frac{\mathfrak{n}_i\,|\boldsymbol{M}_i|}{\mathcal{P}(\omega_i)}\right)^{\frac{1}{\mathrm{d}}}$ are considered as scaling factors, then the sample point $\boldsymbol{x}$ is assigned to class ω_0 if the scaled distance to the kth point from class ω_0 is less than the scaled distance to the kth point from class ω_1; otherwise it is assigned to class ω_1. The *kNN* discriminant function $h_{kNN}(\boldsymbol{x})$ is piecewise linear and therefore continuous. For a problem with two measurements per sample, the decision boundary $h_{1NN}(\boldsymbol{x}) = \tau_{1NN}$ for a 1*NN* classifier is shown in Figure 5. The classification error $\hat{\epsilon}_{kNN}$ for the *kNN* classifier can be computed by substituting Equations (20) and (25) into Equation (14). The resulting expression is a function of both the neighborhood size k and the number of samples in each class $\mathfrak{n}_0$ and $\mathfrak{n}_1$. One reason for using the *kNN* classifier is the following theorem.

Theorem 4 (Devroye *et al.* (1996)). *Given that $\epsilon_{\mathfrak{n}}$ is the classification error for a kNN classifier designed using $\mathfrak{n}$ sample pairs from the set $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$. If $k \to \infty$ and $\mathfrak{n} \to \infty$ in such a way that $\frac{k}{\mathfrak{n}} \to 0$, then for every $\delta > 0$ and every distribution of the pair $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$ there exists an $\mathfrak{n}_m(k)$ such that for $\mathfrak{n} > \mathfrak{n}_m(k)$*

$$\mathcal{P}\big(\epsilon_{\mathfrak{n}} - \epsilon^* > \delta\big) \le 2\,e^{-\left(\frac{\mathfrak{n}\,\delta^2}{72\,\beta^2(\mathrm{d})}\right)},$$

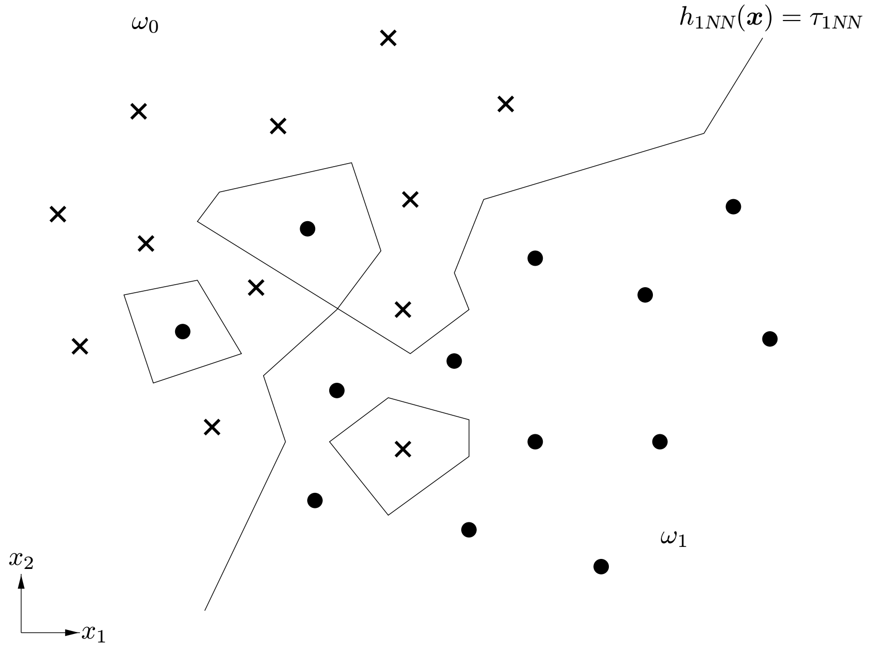

Figure 5: The decision boundary for a 1*NN* classifier for a problem with two measurements per sample.

where $\beta(\mathrm{d})$ *is a constant which depends only on the measurement dimension* d*. Clearly this implies that* $\lim_{\mathrm{n}\to\infty} \mathcal{P}\big(\epsilon_{\mathrm{n}} - \epsilon^* > \delta\big) = 0$*. Thus the kNN classifier is* strongly universally consistent*.*

This result means that as the neighborhood size and the number of samples in the design set $\mathcal{S}_d$ is increased, the classification error for the *kNN* classifier gets arbitrarily close to the Bayes error for every set $\mathcal{S}_d$ except for a group of sets $\mathcal{S}_d$ which have zero probability of being chosen, no matter how the samples in $\mathcal{S}_d$ are selected. A useful property of *kNN* classifiers for finite k is shown by the following theorem.

Theorem 5 (Devroye *et al.* (1996)). *Given that* ϵ_{n} *is the classification error for a kNN classifier designed using* n *sample pairs from the set* $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$*. If* k *is fixed and* $\mathrm{n} \to \infty$ *(which implies that* $\frac{k}{\mathrm{n}} \to 0$*), then for any distribution of the pair* $(\mathcal{X}_\Omega, \mathcal{Y}_\Omega)$

$$\lim_{\mathrm{n}\to\infty} \mathcal{E}(\epsilon_{\mathrm{n}}) = \epsilon^*_{kNN}.$$

Furthermore $\epsilon^* \le \cdots \le \epsilon^*_{kNN} \le \cdots \le \epsilon^*_{2NN} \le \epsilon^*_{1NN} \le 2\,\epsilon^*$*. For a fixed value of* k *the kNN error is bounded by* $\epsilon^*_{kNN} \le \epsilon^* + \frac{1}{\sqrt{(2\,k-1)\,e}}$*.*

This means that for every neighborhood size k, as the number of samples in $\mathcal{S}_d$ is increased, the classification error becomes arbitrarily close to its ideal value, and this ideal value lies between the Bayes error and twice the Bayes error. Note that both of these results assume that all of the samples from $\mathcal{S}_d$ are statistically independent. Also note that these results are independent of the metric used to measure distance.

Unfortunately, the above results apply only in an asymptotic sense; therefore for finite values of k, $\mathfrak{n}_0$, and $\mathfrak{n}_1$ the classification error estimates $\hat{\epsilon}_{kNN}$ for *kNN* still have a bias and variance as given by Equations (18), (19), (15), and (16). The variance can be reduced by increasing $\mathfrak{n}$, and/or by averaging results over several bootstrapped sample sets. The bias can be controlled in part by adjusting the parameters of the *kNN* classifier (e.g. the metric matrices and τ). When the bias is small it can be approximated by the following expression (obtained using a variation on the development in (Fukunaga, 1990)).

$$\begin{aligned}\mathcal{B}_d(\epsilon_{\mathfrak{n}}) \cong\ & c_0\left(\tfrac{1}{k-1}+\tfrac{\Delta t^2}{2}\right)+(c_1+c_2\Delta t)\left(\tfrac{1}{k-2}\right)\left(\tfrac{k-2}{\mathfrak{n}'}\right)^{2/d}+ \\ & (c_3+c_4\Delta t)\left(1+\tfrac{1}{k-2}\right)\left(\tfrac{k-2}{\mathfrak{n}'}\right)^{4/d}-c_5\left(\tfrac{k-1}{\mathfrak{n}'}\right)^{4/d}-c_2\Delta t\left(\tfrac{k-1}{\mathfrak{n}'}\right)^{2/d}\end{aligned} \tag{26}$$

where $\mathfrak{n}' = \mathfrak{n}_0 = \mathfrak{n}_1$, the c_i are constants for a given problem, and Δt is a measure of the deviation of τ from its ideal value in (25). Note that it may be possible to reduce the bias by choosing $\Delta t \neq 0$, which is an idea we exploit later.

5 Summary of Fukunaga's Work

Using the framework established in Sections 1, 2, 3, and 4, a technique for estimating the Bayes error from empirical data may be outlined. It should be stated up front that the following negative result has been proved.

Theorem 6 (Devroye *et al.* (1996)). *For every sample size $\mathfrak{n}$, for any estimate $\hat{\epsilon}_{\mathfrak{n}}$ of the Bayes error ϵ^*, and for every $\delta > 0$, there exists a distribution of samples $(\mathcal{X}, \mathcal{Y})$ such that*

$$\mathcal{E}\big(|\hat{\epsilon}_{\mathfrak{n}} - \epsilon^*|\big) \geq \frac{1}{4} - \delta.$$

This means that *no* method of estimating the Bayes error guarantees a certain finite sample performance for all distributions. This problem exists because for any estimation procedure there is always some sample distribution for which the method converges to the Bayes error arbitrarily slowly with increasing sample size. However having some idea of the best possible classification error for a problem is so useful that we will assert that trying to estimate this quantity is still extremely worthwhile even though it cannot be done well in all cases.

Finite sample estimates of the classification error for the *kNN* classifier in Equation (25) can be obtained using the R and L methods discussed in Section 3. A straightforward implementation of these methods is shown in Algorithm 1. Note that error estimates are computed for $k = 2, 3, ..., k_{max}$ where k_{max} is a user specified parameter. The `KnnDistance` routine returns a list of squared distances from $\boldsymbol{x}[i]$ to its k_{max} nearest neighbors. A brute force implementation of this routine runs in time proportional to $k_{max}\mathfrak{n}\mathsf{d}^2$. There is a significant body of research devoted to the development of more efficient methods for retrieving nearest neighbors, but such enhancements have not been considered in this paper. The overall run time of Algorithm 1 is proportional to $k_{max}\mathfrak{n}^2\mathsf{d}^2$.

In practice, with finite k and $\mathfrak{n}$, the error $\epsilon_{\mathfrak{n}}$ of the volumetric *kNN* method can be much larger than ϵ^*, and the estimates produced by Algorithm 1 may be biased away from ϵ^*. To

Algorithm 1 `ErrKnnBase`: Baseline algorithm for producing R and L error estimates

```
INPUTS: $\mathcal{P}(\omega_0), \mathcal{P}(\omega_1), \mathbf{\Sigma}_0, \mathbf{\Sigma}_1, k_{max}, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Error Counts $e_k$ for $k = 2, 3, ..., k_{max}$

{Initialization}
$\tau = \left(\frac{\mathcal{P}(\omega_0)\mathrm{n}_1|\mathbf{\Sigma}_1|^{1/2}}{\mathcal{P}(\omega_1)\mathrm{n}_0|\mathbf{\Sigma}_0|^{1/2}}\right)^{2/d}$
for $k = 2$ to $k_{max}$ do
  $e_k = 0$
end for

{Main Loop}
for all $(\boldsymbol{x}[i], y[i]) \in \mathcal{S}$ do
  if (Method = L) then
    $D_0^2 \leftarrow$ KnnDistance($\boldsymbol{x}[i], k_{max}, \mathcal{X}_{\omega_0} - \{x[i]\}, \mathbf{\Sigma}_0$)
    $D_1^2 \leftarrow$ KnnDistance($\boldsymbol{x}[i], k_{max}, \mathcal{X}_{\omega_1} - \{x[i]\}, \mathbf{\Sigma}_1$)
  else
    $D_0^2 \leftarrow$ KnnDistance($\boldsymbol{x}[i], k_{max}, \mathcal{X}_{\omega_0}, \mathbf{\Sigma}_0$)
    $D_1^2 \leftarrow$ KnnDistance($\boldsymbol{x}[i], k_{max}, \mathcal{X}_{\omega_1}, \mathbf{\Sigma}_1$)
  end if
  for $k = 2$ to $k_{max}$ do
    $h \leftarrow D_0^2[k]/D_1^2[k]$
    if ( $((h < \tau) \wedge (y[i] = \omega_1)) \ \vee \ ((h > \tau) \wedge (y[i] = \omega_0))$ ) then
      $e_k \leftarrow e_k + 1$
    end if
  end for
end for

Return($\{e_k\}$)
```

obtain a reliable estimate of ϵ^* we must either reduce the bias, adjust for the bias, or both. The bias can often be reduced by modifying the volumetric *kNN* classifier (i.e. changing its parameters). To adjust for the bias, we simply subtract it from ϵ_{n} to produce an estimate of ϵ^*. This requires an analytical expression for the bias as a function of known quantities, or quantities that can be reliably estimated. Fukunaga has developed a family of techniques for error estimation that combine these two approaches (Fukunaga, 1990). The next section discusses bias reduction methods.

In the development that follows, we consider several alternatives to Algorithm 1, some of which make repeated references to nearest neighbors and their distances. To avoid searching the entire data set each time a nearest neighbor reference is made, we pre-compute lists of nearest neighbors and their distances (sorted by distance). There are two lists for each sample, one for nearest neighbors from ω_0 and the other for nearest neighbors from ω_1. For obvious reasons these lists must be at least k_{max} in size. Later we develop techniques that require them to be slightly larger, so let us assume that they have size $LSize$ which satisfies $k_{max} < LSize < 2k_{max}$.

The algorithm to build the nearest neighbor lists (which are referred to collectively as the NN table) is shown in Algorithm 2. The `NNInsert` routine inserts the $(\boldsymbol{x}[j], D^2)$ pair into a (sorted) nearest neighbor list of size $LSize$ for sample $\boldsymbol{x}[i]$. Samples that are farther than $LSize$ away are dropped from the list. The insertion takes time $O(LSize)$. The overall run time of Algorithm 2 is of order $\mathfrak{n}^2(\mathrm{d}^2 + 2k_{max})$.

Algorithm 2 `BuildKnnTable`: Build NN_0 and NN_1 list for each sample (lists contain nearest neighbors and their distances)

```
INPUTS: $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, LSize, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Table of NN Lists $NN_0[\mathfrak{n}][LSize]$ and $NN_1[\mathfrak{n}][LSize]$ where $\mathfrak{n} = |\mathcal{S}|$

for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
  for all $\boldsymbol{x}[j] \in \mathcal{X}_\Omega$ do
    if $((Method = \mathsf{L}) \wedge (\boldsymbol{x}[i] = \boldsymbol{x}[j]))$ then
      Skip to the next $\boldsymbol{x}[j]$
    end if
    if $(y[j] = \omega_0)$ then
      $D^2 \leftarrow \mathcal{D}_0^2(\boldsymbol{x}[i], \boldsymbol{x}[j])$
      NNInsert$(\boldsymbol{x}[j], D^2, NN_0[i])$
    else if $(y[j] = \omega_1)$ then
      $D^2 \leftarrow \mathcal{D}_1^2(\boldsymbol{x}[i], \boldsymbol{x}[j])$
      NNInsert$(\boldsymbol{x}[j], D^2, NN_1[i])$
    end if
  end for
end for

Return$(NN_0, NN_1)$
```

5.1 Bias Reduction Methods

One of the simplest (and most effective) ways of reducing bias is to replace the fixed formula for τ in Algorithm 1 with a data driven choice of $\hat{\tau}$. Note that once we introduce a data driven parameter into the classification rule, we must revise the L estimation procedure. This revision often gives rise to a significant increase in computational complexity. Fukunaga suggests approximation procedures that strike a balance between the ideal procedure and its computational requirements. These procedures often reduce the bias to the point where the error of the *kNN* classifier is quite close to the Bayes error so that the upper and lower bounds obtained using the L and R procedures often provide reasonable bounds for ϵ^*. Fukunaga suggests the following threshold adjustment methods:

ErrKnnTemin: This method chooses a different threshold for each value of k. Each threshold is chosen to minimize the empirical error as shown in Algorithm 3. Once the *kNN* distance ratios have been computed for each sample, the $(h[i], y[i])$ pairs are sorted by their $h[i]$ values so that the `MinError` routine can determine the minimum error with a single scan of the list, updating the error count as it goes. Once the nearest neighbor tables have been built, the run time of this algorithm is proportional to $k_{max}\mathfrak{n}(\log \mathfrak{n}+2)$. This approach is likely to bias the error estimate low, and is therefore appropriate for the R error estimate only.

Algorithm 3 `ErrKnnTemin`: Compute L and R error estimates. Use a different threshold for each value of k. The threshold minimizes the error over the sample set ratios.

```
INPUTS: $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, k_{max}, LSize, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Error Counts $e_k$,   $k = 2, 3, ..., k_{max}$

$(NN_0, NN_1) \leftarrow$ BuildKnnTables($\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, LSize, \mathcal{S}$)

{The $D^2(NN_j[i][k])$ operation retrieves the squared distance to the $k^{th}$ NN of $\boldsymbol{x}[i]$ from $\omega_j$}
for $k = 2$ to $k_{max}$ do
  for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
    $h[i] \leftarrow D^2(NN_0[i][k])/D^2(NN_1[i][k])$
  end for
  $\rho \leftarrow$ Sorth($\{h[i], y[i]\}$) {sort the $(h, y)$ pairs by ratio value}
  $e_k \leftarrow$ MinError($\rho$) {scan the $(h, y)$ pairs to locate the minimum error split}
end for

Return($\{e_k\}$)
```

ErrKnnTlnnlist: This method chooses a different threshold for each data sample (and each value of k). The threshold for sample $\boldsymbol{x}[i]$ is found by leaving $\boldsymbol{x}[i]$ out of the nearest neighbor lists, and then minimizing the empirical error over the remaining samples as shown in Algorithm 4. This is a true *leave-one-out* method in that it completely removes the effect of $\boldsymbol{x}[i]$ when choosing the threshold for $\boldsymbol{x}[i]$ and is therefore used for the L error estimate only. Once the nearest neighbor tables have been built, the run time of this algorithm is proportional to $k_{max}\mathfrak{n}^2(\log \mathfrak{n} + 3)$ (approximately $\mathfrak{n}$ times slower than the **ErrKnnTemin** method).

ErrKnnTlratio: Like the previous method, this method chooses a different threshold for each data sample (and each value of k). It can be viewed as an approximation to the **ErrKnnTlnnlist** method (and is therefore be used for the L error estimate only). Here the threshold for sample $\boldsymbol{x}[i]$ is found by removing its entry from the ratio list and then minimizing the empirical error over the remaining samples. Note that this is not a true leave-one-out method, since $\boldsymbol{x}[i]$ continues to influence its own threshold through its participation in the nearest neighbor lists of the remaining samples. Nevertheless, this method tends to give results comparable to the **ErrKnnTlnnlist** method, and can be implemented more efficiently. A brute force implementation of this method is shown in Algorithm 5. Once the nearest neighbor tables have been built, the run time of this algorithm is proportional to $k_{max}\mathfrak{n}(\mathfrak{n}+\log \mathfrak{n}+1)$, which is roughly a factor of $\log \mathfrak{n}$ faster than Algorithm 4. It would be hard to justify this approximation to `ErrKnnTlnnlist` with such a meager computational savings, but it turns out that a more efficient im-

Algorithm 4 `ErrKnnTlnnlist`: Compute L error estimate. Use a different threshold for each sample. The threshold minimizes the error over the remaining samples with the current sample left out of their NN lists.

```
INPUTS: $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, k_{max}, LSize, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Error Counts $e_k$,   $k = 2, 3, ..., k_{max}$

$(NN_0, NN_1) \leftarrow$ BuildKnnTables$(\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, LSize, \mathcal{S})$

for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
  {Remove $\boldsymbol{x}[i]$ from the NN lists}
  if $(y[i] = \omega_0)$ then
    for $j = 1$ to $\mathfrak{n}$ do
      $NN_0^*[j] \leftarrow$ Remove$(\boldsymbol{x}[i], NN_0[j])$
    end for
  else
    for $j = 1$ to $\mathfrak{n}$ do
      $NN_1^*[j] \leftarrow$ Remove$(\boldsymbol{x}[i], NN_1[j])$
    end for
  end if
  {Find threshold over $\mathcal{X}_\Omega - \{\boldsymbol{x}[i]\}$ and apply to $\boldsymbol{x}[i]$}
  for $k = 2$ to $k_{max}$ do
    for all $\boldsymbol{x}[j] \in \mathcal{X}_\Omega - \{\boldsymbol{x}[i]\}$ do
      $h[j] \leftarrow D^2(NN_0^*[j][k])/D^2(NN_1^*[j][k])$
    end for
    $\rho \leftarrow$ Sorth$(\{h[j], y[j]\})$ {sort the $(h, y)$ pairs by ratio value}
    $\tau \leftarrow$ MinErrorThreshold$(\rho)$
    if ( $((h[i] < \tau) \wedge (y[i] = \omega_1))$ $\vee$ $((h[i] > \tau) \wedge (y[i] = \omega_0))$ ) then
      $e_k \leftarrow e_k + 1$
    end if
  end for
end for

Return$(\{e_k\})$
```

plementation exists with run time proportional to $k_{max}\mathfrak{n}(\log \mathfrak{n} + 4)$ (once the nearest neighbor tables have been built). This saves a factor of $\mathfrak{n}$ over the **ErrKnnTlnnlist** method, and is roughly proportional to the run time of the **ErrKnnTemin** method. This implementation is described in detail in Section 7.

The threshold selection methods just described are very effective at moving the classification performance of the volumetric *kNN* classifier close to the Bayes error, as we will illustrate in Section 8. But our development of these methods is not yet complete. Up to now we have assumed that the metric matrices $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ used in the distance formulas are known in advance. In practice they must be estimated from the data. It is natural to think of $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ as covariance matrices for ω_0 and ω_1, and to estimate them using the well-known maximum

Algorithm 5 `ErrKnnTlratio`: Compute L error estimate. Use a different threshold for each sample. The threshold minimizes the error over the remaining samples with the current sample left out of the ratio list.

```
INPUTS: $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, k_{max}, LSize, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Error Counts $e_k, \quad k = 2, 3, ..., k_{max}$

$(NN_0, NN_1) \leftarrow$ BuildKnnTables($\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, LSize, \mathcal{S}$)

{The $D^2(NN_j[i][k])$ operation retrieves the squared distance to the $k^{th}$ NN of $\boldsymbol{x}[i]$ from $\omega_j$}
for $k = 2$ to $k_{max}$ do
  for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
    $h[i] \leftarrow D^2(NN_0[i][k])/D^2(NN_1[i][k])$
  end for
  $\rho \leftarrow$ Sorth($\{h[i], y[i]\}$) {sort the $(h, y)$ pairs by ratio value}
  for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
    $\tau \leftarrow$ MinErrorThreshold($\rho - \{(h[i], y[i])\}$)
    if ( $((h[i] < \tau) \wedge (y[i] = \omega_1)) \vee ((h[i] > \tau) \wedge (y[i] = \omega_0))$ ) then
      $e_k \leftarrow e_k + 1$
    end if
  end for
end for

Return($\{e_k\}$)
```

likelihood formula from Equation (22). However, this choice for $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ does not necessarily produce an optimal classifier. Fukunaga's analysis suggests that a different choice of $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ may reduce the bias of the classifier (Fukunaga, 1990). Unfortunately there is no closed form expression for the optimal $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$, so in the interest of simplicity (and computational efficiency) we use the expression in (22) (as does Fukunaga).

Since $\hat{\mathbf{\Sigma}}_0$ and $\hat{\mathbf{\Sigma}}_1$ are data dependent parameters of the classifier, they will contribute to the bias and variance of our error estimates. Specifically, when $\hat{\mathbf{\Sigma}}_j$ is used in the classification of $\boldsymbol{x}_j[i]$, the result is likely to be unfairly biased in favor of assigning $\boldsymbol{x}_j[i]$ to ω_j. This is acceptable for the R error estimates, but not for the L error estimates. To remove this bias for the L error estimates we must remove the contribution of $\boldsymbol{x}_j[i]$ on $\hat{\mathbf{\Sigma}}_j$ before forming its nearest neighbor list $NN_j[i]$.

Formation of the nearest neighbor lists is currently the computational bottleneck in our algorithms, and we would like to avoid computing them twice: once for the R error estimates and a second time for the L error estimates. Our solution is as follows. First we build the nearest neighbor lists using Algorithm 2 with metric matrices $\hat{\mathbf{\Sigma}}_0$ and $\hat{\mathbf{\Sigma}}_1$ formed from all $\mathfrak{n}$ data samples. These lists are then used to compute the R error estimates. Afterwards we *adjust* the nearest neighbor lists according to a leave-one-out strategy as follows. For each sample $\boldsymbol{x}_j[i]$: we remove its influence on $\hat{\mathbf{\Sigma}}_j$, recompute the distance to each sample in its nearest neighbor list using the new $\hat{\mathbf{\Sigma}}_j$, and then re-order the neighbors as needed. Once all entries in the nearest neighbor list have been adjusted, $\hat{\boldsymbol{\Sigma}}_j$ is restored to its original form. Note that this operation may cause a re-ordering of the samples in the list. For this reason the original nearest neighbor list should be larger than k_{max} so that samples that have the potential of moving into one of the first k_{max} positions (from outside) are considered in the re-ordering process. The exact size needed for the list is problem dependent. Empirical observations suggest that $LSize = 2k_{max}$ is a very conservative choice. The complete algorithm is shown in Algorithm 6. The run time of this algorithm is proportional to $2\mathsf{n}(\mathsf{d}^2 + 4k_{max}^2 + 2k_{max}\mathsf{d}^2)$, which is usually dominated by the $\mathsf{n}k_{max}\mathsf{d}^2$ term. For $k_{max} \ll \mathsf{n}$ this is nearly n times faster than rebuilding the nearest neighbor lists from scratch.

Algorithm 6 `AdjustNNlistLOO`: Adjust NN_0 and NN_1 list for each sample by removing the effect of $\boldsymbol{x}_j[i]$ on $\hat{\boldsymbol{\Sigma}}_i$ and then recomputing the distance to its neighbors, and re-ordering them in the list.

```
INPUTS: $\boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1, LSize, NN_0[\mathsf{n}][LSize], NN_1[\mathsf{n}][LSize], \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Adjusted NN Lists $NN_0[\mathsf{n}][LSize]$ and $NN_1[\mathsf{n}][LSize]$

for all $\boldsymbol{x}[i] \in \mathcal{X}_{\omega_0}$ do
  $\hat{\boldsymbol{\Sigma}}_0 \leftarrow$ remove effect of $\boldsymbol{x}[i]$ on $\hat{\boldsymbol{\Sigma}}_0$
  for $j = 1$ to $LSize$ do
    $\boldsymbol{x} \leftarrow$ NNRemove($NN_0[i], j$)
    $D^2 \leftarrow \mathcal{D}_0^2(\boldsymbol{x}[i], \boldsymbol{x})$
    NNInsert($\boldsymbol{x}, D^2, NN_0[i]$)
  end for
  $\hat{\boldsymbol{\Sigma}}_0 \leftarrow$ restore effect of $\boldsymbol{x}[i]$ on $\hat{\boldsymbol{\Sigma}}_0$
end for

for all $\boldsymbol{x}[i] \in \mathcal{X}_{\omega_1}$ do
  $\hat{\boldsymbol{\Sigma}}_1 \leftarrow$ remove effect of $\boldsymbol{x}[i]$ on $\hat{\boldsymbol{\Sigma}}_1$
  for $j = 1$ to $LSize$ do
    $\boldsymbol{x} \leftarrow$ NNRemove($NN_1[i], j$)
    $D^2 \leftarrow \mathcal{D}_1^2(\boldsymbol{x}[i], \boldsymbol{x})$
    NNInsert($\boldsymbol{x}, D^2, NN_1[i]$)
  end for
  $\hat{\boldsymbol{\Sigma}}_1 \leftarrow$ restore effect of $\boldsymbol{x}[i]$ on $\hat{\boldsymbol{\Sigma}}_1$
end for

Return($NN_0, NN_1$)
```

When using the **ErrKnnTlratio** method to form the L error estimate, the adjustments made by Algorithm 6 are sufficient to compensate for the use of a data driven $\hat{\boldsymbol{\Sigma}}$. These adjustments can be made prior to invocation of the **ErrKnnTlratio** procedure (but obviously after the nearest neighbor tables have been built and used for the R error estimate). When the **ErrKnnTlnnlist** method is used, however, there are two places where the $\hat{\boldsymbol{\Sigma}}_j$ matrices should be adjusted. When $\boldsymbol{x}_j[i]$ is removed from the nearest neighbor lists, we should also remove its influence on $\hat{\boldsymbol{\Sigma}}_j$ before computing nearest neighbors for the remaining samples.

Routine Name	**Time Complexity**	**Comments**
`BuildKnnTable`	$t_p = \mathfrak{n}^2(\mathrm{d}^2 + 2k_{max})$	pre-processing step
`AdjNNlistLOO`	$t_a = 4\mathfrak{n}k_{max}^2\mathrm{d}^2$	(L) adjusts NN list for each sample by removing effect of sample on metric matrix
`ErrKnnBase`	$k_{max}\mathfrak{n}^2\mathrm{d}^2$	(R or L) baseline routine for estimating ϵ, estimate often biased away from ϵ^*
`ErrKnnTemin`	$t_p +$ $k_{max}\mathfrak{n}(\log\mathfrak{n} + 2)$	(R) choose threshold that minimizes empirical error (biased low)
`ErrKnnTlnnlist`	$t_p + t_a +$ $k_{max}\mathfrak{n}^2(\log\mathfrak{n} + 3)$	(L) choose threshold that minimizes empirical error after leaving $\boldsymbol{x}[i]$ out of NN lists (approx. LOO rule for $\hat{\boldsymbol{\Sigma}}$)
`ErrKnnTlnnlist`*	$t_p +$ $k_{max}^2\mathfrak{n}^2\mathrm{d}^2$	(L) choose threshold that minimizes empirical error after leaving $\boldsymbol{x}[i]$ out of NN lists (exact LOO rule for $\hat{\boldsymbol{\Sigma}}$)
`ErrKnnTlratio`	$t_p + t_a +$ $k_{max}\mathfrak{n}(\mathfrak{n} + \log\mathfrak{n} + 1)$	(L) choose threshold that minimizes empirical error after leaving $h[i]$ out of the Ratio list
`ErrKnnTlratioFast`	$t_p + t_a +$ $k_{max}\mathfrak{n}(\log\mathfrak{n} + 4)$	(L) choose threshold that minimizes empirical error after leaving $h[i]$ out of the Ratio list, FAST version

Table 1: Algorithms for error-bound estimation and their run times.

In addition, when computing the nearest neighbor lists for each the remaining sample $\boldsymbol{x}_q[p]$, we should remove the influence of $\boldsymbol{x}_q[p]$ on $\hat{\boldsymbol{\Sigma}}_q$. Algorithm 6 is of little help in providing an efficient implementation of these changes for the **ErrKnnTlnnlist** method. They must be incorporated directly in the `ErrKnnTlnnlist` procedure, which leads to an algorithm with run time $\Theta(2\mathfrak{n}^2\mathrm{d}^2 + \mathfrak{n}^2k_{max}(4k_{max}\mathrm{d}^2 + \log\mathfrak{n}))$, which simplifies to $\Theta(k_{max}^2\mathrm{d}^2\mathfrak{n}^2)$ when $k_{max}\mathrm{d}^2 = \Omega(\log\mathfrak{n})$. This is significantly slower than the algorithm used to build the original nearest neighbor lists and is unacceptably slow for medium-to-large data sets. For this reason we have implemented the following approximation to this method. The second adjustment (removing the effect of $\boldsymbol{x}[i]$ on $\hat{\boldsymbol{\Sigma}}$ for its own nearest neighbor list) is performed using Algorithm 6, and then the **ErrKnnTlnnlist** method is performed as before using the adjusted nearest neighbor lists. This mirrors our implementation of the **ErrKnnTlratio** method.

Table 5.1 provides a summary of the algorithms we have discussed in this section and their run times.

5.2 Bias Adjustment Methods

This section discusses methods that estimate the Bayes error by explicitly (or implicitly) estimating the bias, and then subtracting it from $\hat{\epsilon}_n$. Our approach requires an analytic expression for the bias. Let $\Delta\epsilon = \epsilon_n - \epsilon^*$ where ϵ_n is the error estimated by the L method and ϵ^* is the Bayes error. Define the bias to be $\bar{\Delta\epsilon} = E_n[\Delta\epsilon]$ where the expectation is over sample sets of size n. Under the assumption that $n_0 = n_1 = n/2$, Fukunaga gives the following (approximate) expression for the bias (Fukunaga, 1990)

$$\bar{\Delta\epsilon} \cong b_1\left(\frac{1}{k}\right) + b_2\left(\frac{k}{n}\right)^{2/d} + b_3\left(\frac{k}{n}\right)^{4/d} \tag{27}$$

where b_1, b_2 and b_3 are constant for a given problem. This equation can be re-written as

$$E_n[\epsilon_n] \cong \epsilon^* + b_1\left(\frac{1}{k}\right) + b_2\left(\frac{k}{n}\right)^{2/d} + b_3\left(\frac{k}{n}\right)^{4/d} \tag{28}$$

which suggests that we can produce a Bayes error estimate by fitting a curve to the L error estimates (as a function of k), and choosing the constant term as our estimate of ϵ^*. To approximate the expected value on the left hand side of (28) we average the error estimates over several runs of the L estimation method using bootstrapped samples.

5.3 Summary of Bias Issues

Let $\Delta\tau = \hat{\tau} - \tau$ where τ is given in Equation (25) and $\hat{\tau}$ is the data driven choice of threshold. A careful study of Funukaga's bias derivation reveals that the coefficients b_2 and b_3 are (nontrivial) functions of $\Delta\tau$, $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$, and that the bias from the $\left(\frac{k}{n}\right)^{2/d}$ and $\left(\frac{k}{n}\right)^{4/d}$ terms can be reduced through the proper choice of these parameters. This observation provides a formal justification of the threshold selection methods in Section 5.1. It also suggests that the bias might be further reduced through proper selection of the metric matrices, but the best way to exploit this knowledge is still an open problem.

In contrast, the curve-fitting approach described in the previous section would seem to alleviate the need for bias reduction, since it compensates for the bias directly. However, preliminary experiments with this approach show that the error estimates are quite poor when the bias is large. There are several reasons for this, not the least of which is that the bias expression in (27) was derived under the assumption that the *kNN* classifier closely approximates the optimal classifier. This assumption is clearly violated when the bias is large.

We believe that the most promising approach to Bayes error estimation is a combination bias reduction and bias adjustment. We revisit this issue in Section 9.

6 Technical Issues not covered by Fukunaga

6.1 Data-Driven Threshold Selection

The threshold selection process employed by the `MinErrorThreshold` routine in Algorithms 4 and 5 requires further explanation. This routine accepts a sorted list of $(h[i], y[i])$ pairs and returns a threshold $\hat{\tau}$ that minimizes the empirical error under the decision rule $h \gtrless_{\omega_0}^{\omega_1} \hat{\tau}$. Since the $(h[i], y[i])$ pairs are sorted, the minimum error position can be determined with a single pass through the list, tracking the error that results from positioning the threshold between each adjacent pair of ratios. But there are two unresolved issues with this routine. The first is how to choose the actual threshold value from the interval of possibilities, and the second is what to do about multiple minima.

We consider first the issue of how to choose the threshold value. Suppose the minimum error position falls between $h[i^*]$ and $h[i^*+1]$. Any threshold in the interval $(h[i^*], h[i^*+1])$ will minimize the empirical error, but the "left-out" sample may be classified differently depending on our choice. In fact, one could argue that this choice of threshold has a strong influence on the accuracy of the leave-one-out error estimate since the classification of the "left-out" sample is less likely to change if it does not fall in this interval. The question is then how to choose the threshold from $(h[i^*], h[i^* + 1])$. The midpoint of the interval would seem to be a likely choice, but we argue that it is not the best choice. The thresholded h value is proportional to the *ratio* of probability functions for ω_0 and ω_1, so its sensitivity is not equal with respect to ω_0 and ω_1. On the other hand, the log-ratio is equally sensitive with respect to ω_0 and ω_1. So a better choice of threshold might be the midpoint of the log ratio interval. Using this choice, and inverting the log function to obtain a distance ratio threshold gives

$$\hat{\tau} = \exp\left(\frac{\log(h[i^*]) + \log(h[i^* + 1])}{2}\right) \tag{29}$$

This formula is currently in use, although its optimality remains open.

We now turn to the second issue. To this point we have assumed that the minimum error occurs at a single position in the list, when in fact it may show up in multiple positions. In fact, we might expect this to be the norm. Once again we can argue that our treatment of this situation may have a strong impact on the accuracy of the leave-one-out error estimate, since the classification of "left-out" samples changes more frequently in the multiple minima case. Our solution is to track *all* minimum error positions and return a threshold for each (using the formula in (29)). Classification of "left-out" samples is then accomplished using a majority vote over the classifications determined from each threshold separately.

Finally, we briefly mention a different strategy for selecting thresholds which simultaneously addresses both issues raised above. The idea is to fit a low-order polynomial (e.g. second order) to the "error count versus ratio" data (or perhaps the "error count versus log-ratio" data), and return the threshold that minimizes this polynomial function. This approach has the potential of both simplifying our implementation and producing more accurate error estimates.

6.2 Incorporating Prior Probabilities

The sample set $\mathcal{X}_\Omega$ determines an empirical distribution of data that is assumed to be a fair representation of the true distribution. Similarly, the relative frequency of samples from $\mathcal{X}_{\omega_0}$ and $\mathcal{X}_{\omega_1}$ determine empirical priors of the form

$$\hat{\mathcal{P}}(\omega_0) = \frac{n_0}{n}, \quad \hat{\mathcal{P}}(\omega_1) = \frac{n_1}{n}$$

that are assumed to be (approximately) correct. It is not uncommon however for the empirical priors to differ significantly from the actual priors. In this case the (data driven) error estimates will also differ significantly from the truth. To remedy this situation, the true priors must be incorporated into the error estimation process. There are two ways of doing this:

Resampling: This method adjusts the empirical priors to match the true priors by changing the relative frequency of samples in $\mathcal{X}_{\omega_0}$ and $\mathcal{X}_{\omega_1}$. This adjustment is accomplished by resampling (with replacement) the data from $\mathcal{X}_{\omega_0}$ and $\mathcal{X}_{\omega_1}$ to produce new sample sets $\tilde{\mathcal{X}}_{\omega_0}$ and $\tilde{\mathcal{X}}_{\omega_1}$ for which $\tilde{n}_0/\tilde{n} = \mathcal{P}(\omega_0)$ and $\tilde{n}_1/\tilde{n} = \mathcal{P}(\omega_1)$.

Data Weighting: This method associates different weights with the data from $\mathcal{X}_{\omega_0}$ and $\mathcal{X}_{\omega_1}$. The weight for ω_i is given by $\alpha_{\omega_i} = \frac{\mathcal{P}(\omega_i)}{\hat{\mathcal{P}}(\omega_i)}$. This weighting has the following affect on the volumetric *kNN* method. To determine the k^{th} nearest neighbor distance to be used in the likelihood ratio we compute the distance to the $m = k/\alpha$ neighbor from the data set. If m is not an integer, then the distance we seek can be approximated by interpolating between the distances to the $\lfloor m \rfloor$ and $\lceil m \rceil$ neighbors.

6.3 Tracking Error Types

The Bayes error is comprised of two parts, ϵ_0 and ϵ_1, which represent the (expected) misclassification rate of data from ω_0 and ω_1 respectively. It is a simple matter to produce estimates of ϵ_0 and ϵ_1 along with the estimates of ϵ. This is accomplished by counting the type 0 and type 1 errors separately. This represents a trivial modification to the existing algorithms. Every time an error count is adjusted its type (0 or 1) is determined by the corresponding sample label.

6.4 Covariance Modes and Their Updates

The metric matrices $\hat{\mathbf{\Sigma}}_0$ and $\hat{\mathbf{\Sigma}}_1$ are currently obtained as covariance matrix estimates from the data in $\mathcal{X}_{\omega_0}$ and $\mathcal{X}_{\omega_1}$ respectively. These matrices represent roughly d^2 free parameters estimated from n independent samples. If n is small relative to the number of free parameters, then the (high variance) matrix estimates may be so poor that they cause a serious degradation in the accuracy of the error estimates produced by the volumetric *kNN* method. To prevent this, we place restrictions on the covariance matrices so that the ratio of n to the number of free parameters is maintained at an acceptable level. As a rule of thumb this ratio should be larger than 1. With this in mind, the covariance matrices are implemented using one of four modes depending on the number of samples available. These modes are summarized in Table 6.4. In the FULL mode $\hat{\mathbf{\Sigma}}$ is estimated using Equation (22). In the DIAGONAL mode $\hat{\mathbf{\Sigma}}$ is estimated using only the diagonal entries from Equation (22) and the off diagonal entries are set to zero. The SCALED IDENTITY mode uses $\hat{\mathbf{\Sigma}} = s\boldsymbol{I}$ where s is a (data driven) scaler and $\boldsymbol{I}$ is the identity matrix. The IDENTITY mode uses $\hat{\mathbf{\Sigma}} = \boldsymbol{I}$ (i.e. the Euclidean distance).

Distance calculations require the *inverse* of these matrices, or more specifically the product of their inverse with a difference vector. In addition, the leave-one-out methods require updates to these matrices that reflect the removal (and restoration) of individual samples. These operations are trivial when one of the last three matrix modes are in use, but this is not the case

Mode	Number of Samples
FULL	$\mathsf{d}^2+\mathsf{d}+1<\mathsf{n}_i$
DIAGONAL	$2\mathsf{d}+1<\mathsf{n}_i\leq\mathsf{d}^2+\mathsf{d}+1$
SCALED IDENTITY	$2+\mathsf{d}<\mathsf{n}_i\leq 2\mathsf{d}+1$
IDENTITY	$1<\mathsf{n}_i\leq 2+\mathsf{d}$

Table 2: Covariance Matrix Modes

with the FULL covariance mode. To guarantee that all operations are implemented efficiently and robustly in the FULL covariance mode the matrices are stored as QR decompositions. In this form it is possible to form an inverse matrix vector product in order d^2 time. In addition, all matrix updates can be performed in order d^2 time.

7 Algorithm Enhancements and Computational Issues

7.1 An Efficient Implementation of the ErrKnnTlratio Method

In section 5.1 we described a brute force implementation of the **ErrKnnTlratio** method of threshold selection. This method chooses a threshold for each sample $\boldsymbol{x}[i]$ by removing $h[i]$ from the ratio list and then minimizing the empirical error over the remaining samples. In this section we describe a more efficient implementation of this method. It starts by computing the sorted ratio list and scanning the entire list to determine the minimum error (as in Algorithm 3). This error is then *adjusted* to compensate for the effect of leaving out samples one at a time. The final procedure is quite simple, although the derivation that follows is somewhat involved.

Suppose we've scanned the ratio list and determined the minimum error e_A and corresponding threshold(s). To aid in our exposition, a hypothetical landscape showing the error count as a function of threshold value is shown in Figure 6. Note that the error count changes by 1 each time the threshold moves past a ratio $h[i]$ from our list. Note also that the landscape contains multiple global minima, a situation we expect to occur frequently in practice. The classification rule that we employ when there are multiple global minima is a majority vote over the classifications obtained using each threshold separately. Ties are resolved by a coin flip. Thus, if $\boldsymbol{x}[i]$ belongs to ω_0 and the number of global minima to the right of $h[i]$ is greater (less) than the number to the left, then $x[i]$ is correctly (incorrectly) classified (prior to leave-one-out). Similarly, if $\boldsymbol{x}[i]$ belongs to ω_1 and the number of global minima to the left of $h[i]$ is greater (less) than the number to the right, then $x[i]$ is correctly (incorrectly) classified (prior to leave-one-out). Samples for which the number of global minima is equal on the left and right are classified correctly (incorrectly) half the time (prior to leave-one-out).

Now, in the leave-one-out procedure the removal of $h[i]$ from the list has the following effect(s) on the landscape:

1. If $\boldsymbol{x}[i] \in \omega_0$ then the landscape is *increased by 1 at all positions to the right* of $h[i]$ (see illustration in Figure 7).

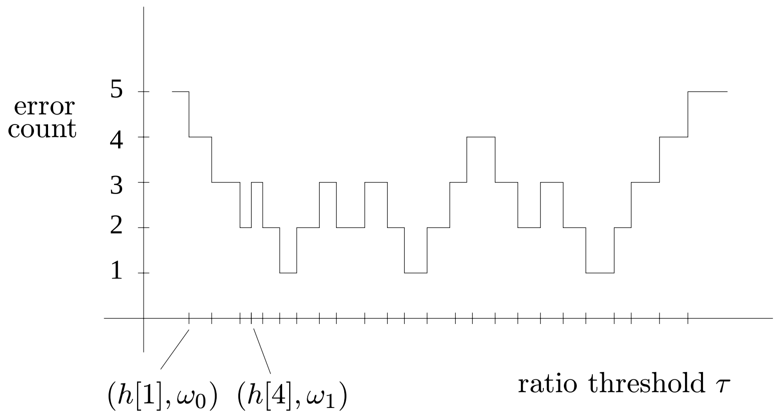

Figure 6: Landscape of error count versus threshold value.

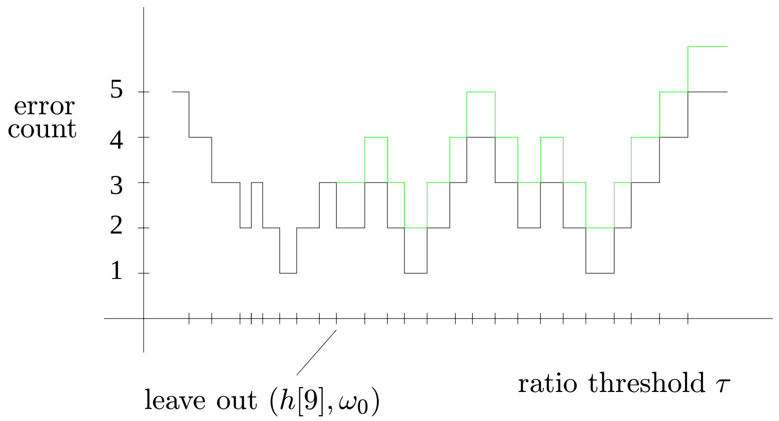

Figure 7: Adjustment to landscape as a result of leaving out $(h[9], \omega_0)$. Adjusted landscape is shown as a dashed line.

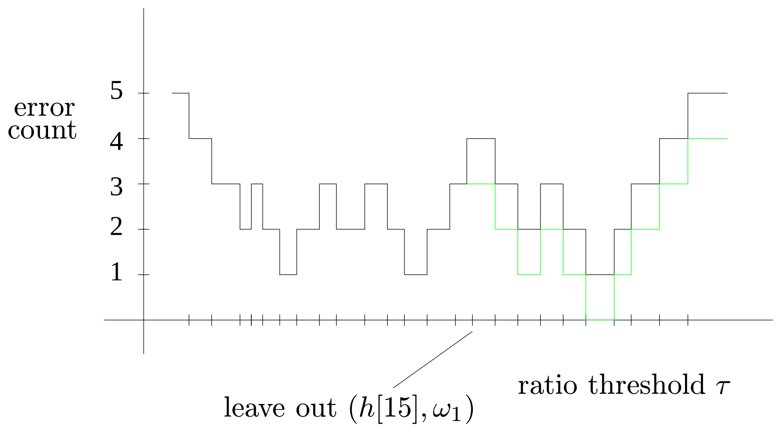

Figure 8: Adjustment to landscape as a result of leaving out $(h[15], \omega_1)$. Adjusted landscape is shown as a dashed line.

2. If $\boldsymbol{x}[i] \in \omega_1$ then the landscape is *decreased by 1 at all positions to the right* of $h[i]$ (see illustration in Figure 8).

The manner in which these changes affect the (re-)classification of the "left-out" sample can easily be determined as a function of the following variables:

$NLeft_A[i]$: The number of *global* minima to the *left* of ratio $h[i]$.

$NRight_A[i]$: The number of *global* minima to the *right* of ratio $h[i]$.

$NLeft_B[i]$: The number of *local* minima with error $e_B = e_A + 1$ to the *left* of ratio $h[i]$.

$NRight_B[i]$: The number of *local* minima with error $e_B = e_A + 1$ to the *right* of ratio $h[i]$.

The following cases represent the possible ways in which the classification of the "left-out" sample can change. We consider first the cases for which $\boldsymbol{x}[i] \in \omega_0$.

Case 1: Prior to being left out $\boldsymbol{x}[i]$ is classified correctly and there are one or more global minima to the left of $h[i]$ (i.e. $NRight_A[i] > NLeft_A[i] > 0$). After its removal, the global minima on the right will increase by 1, and the new classification will be determined entirely by the global minima on the left, which all vote to assign $\boldsymbol{x}[i]$ to ω_1. Thus, $\boldsymbol{x}[i]$ is now misclassified, and the error count should be increased by 1.

Case 2: Prior to being left out $\boldsymbol{x}[i]$ is classified correctly, and there are no global minima on the left (i.e. $NRight_A[i] > NLeft_A[i] = 0$). After its removal, the global minima on the right increase by 1 and end up competing with the *local* minima on the left to classify $\boldsymbol{x}[i]$. If $NLeft_B[i] > NRight_A[i]$ then $\boldsymbol{x}[i]$ will be misclassified, and the error count should be increased by 1. If $NLeft_B[i] = NRight_A[i]$ then there is a tie, and the error count should be increased by 1/2. If $NLeft_B[i] < NRight_A[i]$ then $\boldsymbol{x}[i]$ remains correctly classified, and no adjustment in the error count is needed.

Case 3: Prior to being left out, $\boldsymbol{x}[i]$ is classified correctly 1/2 of the time (i.e. $NRight_A[i] = NLeft_A[i] \neq 0$). After its removal, the global minima on the right will increase by 1, and the new classification will be determined entirely by the global minima on the left, which all vote to assign $\boldsymbol{x}[i]$ to ω_1. Thus, $\boldsymbol{x}[i]$ is now misclassified, and the error count should be increased by 1/2.

Note that if $\boldsymbol{x}[i] \in \omega_0$ is *misclassified* prior to being left out, then raising the landscape to the right of $h[i]$ can *never* correct the classification of $\boldsymbol{x}[i]$. So the error count will never change as a result of leaving out a sample that is already misclassified. Thus, the cases enumerated above represent *all* possible ways in which the error count can change when $\boldsymbol{x}[i] \in \omega_0$. Note that all of them lead to an *increase* in the error count (that is, the error count can never decrease as a result of the leave-one-out adjustment). The ways in which the error count can change when $\boldsymbol{x}[i] \in \omega_1$ are symmetric with those above, and are omitted for brevity.

We are now ready to describe the fast implementation of the **ErrKnnTlratio** method. The algorithm is shown in Algorithm 7. Once the sorted ratio list is formed, it is scanned three times. The first scan is performed by the `LocateGlobalMin` routine which determines the number, locations, and value, (N_A, I_A, and e_A), of the global minima [1]. The second scan is performed by the `LocateLocalMin1` routine which determines the number and locations, (N_A, I_B), of the local minima with error value $e_A + 1$. Finally, this information is passed to the `AdjustErr` routine that scans the ratio list one last time, adjusting the error count for each sample using the rules outlined above. One final observation is in order. The classification error will never change for samples whose ratios fall entirely to the left or right of the minima found in the first two scans. So the `AdjustErr` routine can restrict its scan to the subset of ratios that fall in this range. Since this subset is often a small fraction of the total it is computationally advantageous to do so. Once the nearest neighbor tables have been built, the run time of this algorithm is proportional to $\mathfrak{n}k_{max}(\log \mathfrak{n} + 4)$.

Algorithm 7 `ErrKnnTlratioFast`: Compute `L` error estimate. Use a different threshold for each sample. The threshold minimizes the error over the remaining samples with the current sample left out of the ratio list. This is the FAST implementation of this method.

```
INPUTS: $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, k_{max}, LSize, \mathcal{S} = \mathcal{X}_\Omega \times \mathcal{Y}_\Omega$
OUTPUTS: Error Counts $e_k$,   $k = 2, 3, ..., k_{max}$

$(NN_0, NN_1) \leftarrow$ BuildKnnTables($\mathbf{\Sigma}_0, \mathbf{\Sigma}_1, LSize, \mathcal{S}$)

{The $D^2(NN_j[i][k])$ operation retrieves the squared distance to the $k^{th}$ NN of $\boldsymbol{x}[i]$ from $\omega_j$}
for $k = 2$ to $k_{max}$ do
  for all $\boldsymbol{x}[i] \in \mathcal{X}_\Omega$ do
    $h[i] \leftarrow D^2(NN_0[i][k])/D^2(NN_1[i][k])$
  end for
  $\rho \leftarrow$ Sorth({$h[i], y[i]$}) {sort the $(h, y)$ pairs by ratio value}
  $(I_A, N_A, e_A) \leftarrow$ LocateGlobalMin($\rho$)
  $(I_B, N_B) \leftarrow$ LocateLocalMin1($\rho, e_A$)
  $e_k \leftarrow$ AdjustErr($\rho, e_A, I_A, N_A, I_B, N_B$)
end for

Return({$e_k$})
```

[1]Note that the locations are represented by a list of indices I_A that can be used to index the ratios $h[i]$ where the minima occur.

Algorithm 8 `AdjustErr`: Scan a subset of the ratio list and adjust the error count to account for additional errors due to the leave-one-out procedure.

```
INPUTS: $\rho$, $e_A$, $I_A$, $I_B$, $N_A$, $N_B$
OUTPUTS: Adjusted Error Count $e_A$

$i_{start} \leftarrow \texttt{min}(I_A[1], I_1[1])$
$i_{end} \leftarrow \texttt{max}(I_A[N_A], I_1[N_1])$
$NLeft_A \leftarrow 0$ , $NLeft_B \leftarrow 0$
$NRight_A \leftarrow N_A$ , $NRight_B \leftarrow N_1$
for $i \leftarrow i_{start}$ to $i_{end}$ do
  if $(y[i] = \omega_A)$ then
    if $(NLeft_A < NRight_A)$ then
      if $(NLeft_A > 0)$ then
        $e_A \leftarrow e_A + 1$
      else if $(NLeft_B > NRight_A)$ then
        $e_A \leftarrow e_A + 1$
      else if $(NLeft_B = NRight_A)$ then
        $e_A \leftarrow e_A + 1/2$
      end if
    else if $(NLeft_A = NRight_A)$ then
      $e_A \leftarrow e_A + 1/2$
    end if
  else
    if $(NLeft_A > NRight_A)$ then
      if $(NRight_A > 0)$ then
        $e_A \leftarrow e_A + 1$
      else if $(NRight_B > NLeft_A)$ then
        $e_A \leftarrow e_A + 1$
      else if $(NRightB = NLeft_A)$ then
        $e_A \leftarrow e_A + 1/2$
      end if
    else if $(NLeft_A = NRight_A)$ then
      $e_A \leftarrow e_A + 1/2$
    end if
  end if
  if $(I_A[NLeft_A] = i)$ then
    $NLeft_A \leftarrow NLeft_A + 1$
    $NRight_A \leftarrow NRight_A - 1$
  end if
  if $(I_B[NLeft_B] = i)$ then
    $NLeft_B \leftarrow NLeft_B + 1$
    $NRight_B \leftarrow NRight_B - 1$
  end if
end for

Return($e_A$)
```

7.2 Improved Efficiency by Combining Threshold Options

The true `L` error estimation procedure `ErrKnnTlnnlist` is computationally expensive. We have introduced the `ErrKnnTlratioFast` procedure as a more efficient means of achieving similar results. For a slight increase in computational effort, however, it may be possible to achieve nearly identical results to `ErrKnnTlnnlist`. This idea is simple. We first run the `ErrKnnTlratioFast` procedure and then run the more expensive `ErrKnnTlnnlist` procedure, but only on those samples whose leave-one-out classification is likely to change. Samples whose classification is most likely to change fall near the decision boundary. They can be identified quite easily during our run of the `ErrKnnTlratioFast` routine. As it makes its second pass through the ratio list the `LocateLocalMin1` subroutine simply tags all samples whose error count is close to e_A. In many problems these samples make up only a small fraction of the total, so the additional computational overhead of running the `ErrKnnTlnnlist` on them is small.

8 Experimental Results

This section presents empirical results obtained by applying the error estimation techniques to the three benchmark problems from (Fukunaga, 1990). All three of the benchmarks are two-class problems in $\mathtt{d} = 8$ dimensions with Gaussian distributions for both ω_0 and ω_1. The Bayes error can be determined analytically for these problems and is specified along with the problem definition below. The acronyms $\boldsymbol{I}$-$\boldsymbol{I}$, $\boldsymbol{I}$-$4\boldsymbol{I}$ and $\boldsymbol{I}$-$\boldsymbol{\Lambda}$, are due to Fukunaga.

Problem $\boldsymbol{I}$-$\boldsymbol{I}$:

$\boldsymbol{\mu}_0^T = [0, 0, ..., 0]$, $\boldsymbol{\mu}_1^T = [2.56, 0, ..., 0]$,
$\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{I}$,
Bayes Error = 0.10

Problem $\boldsymbol{I}$-$4\boldsymbol{I}$:

$\boldsymbol{\mu}_0^T = \boldsymbol{\mu}_1^T = [0, 0, ..., 0]$,
$\boldsymbol{\Sigma}_0 = \boldsymbol{I}$, $\boldsymbol{\Sigma}_1 = 4\boldsymbol{I}$,
Bayes Error = 0.09

Problem $\boldsymbol{I}$-$\boldsymbol{\Lambda}$:

$\boldsymbol{\mu}_0^T = [0, 0, ..., 0]$, $\boldsymbol{\mu}_1^T = [3.86, 3.10, 0.84, 0.84, 1.64, 1.08, 0.26, 0.01]$
$\boldsymbol{\Sigma}_0 = \boldsymbol{I}$, $\boldsymbol{\Sigma}_1 = Diag(8.41, 12.06, 0.12, 0.22, 1.49, 1.77, 0.35, 2.73)$,
Bayes Error = 0.019

The error estimation techniques discussed in Section 5 were applied all three problems. The first set of results is shown in Figures 9-11, which plot the error estimates versus k for $k = 2, 3, ..., 30$. These results were obtained using $\mathtt{n}_0 = \mathtt{n}_1 = 100$, $\hat{\boldsymbol{\Sigma}}_i = \boldsymbol{\Sigma}_i$ (i.e. true covariance matrices), and averaging the error estimates over 10 independent sample sets. These conditions are identical to those of Experiment 14, pp. 349-350 in (Fukunaga, 1990), and our results are a very close match, with the exception of the L error estimates for the $\boldsymbol{I}$-$\boldsymbol{\Lambda}$ problem. We believe the result in (Fukunaga, 1990) to be in error. This is based partly on the similarity of our

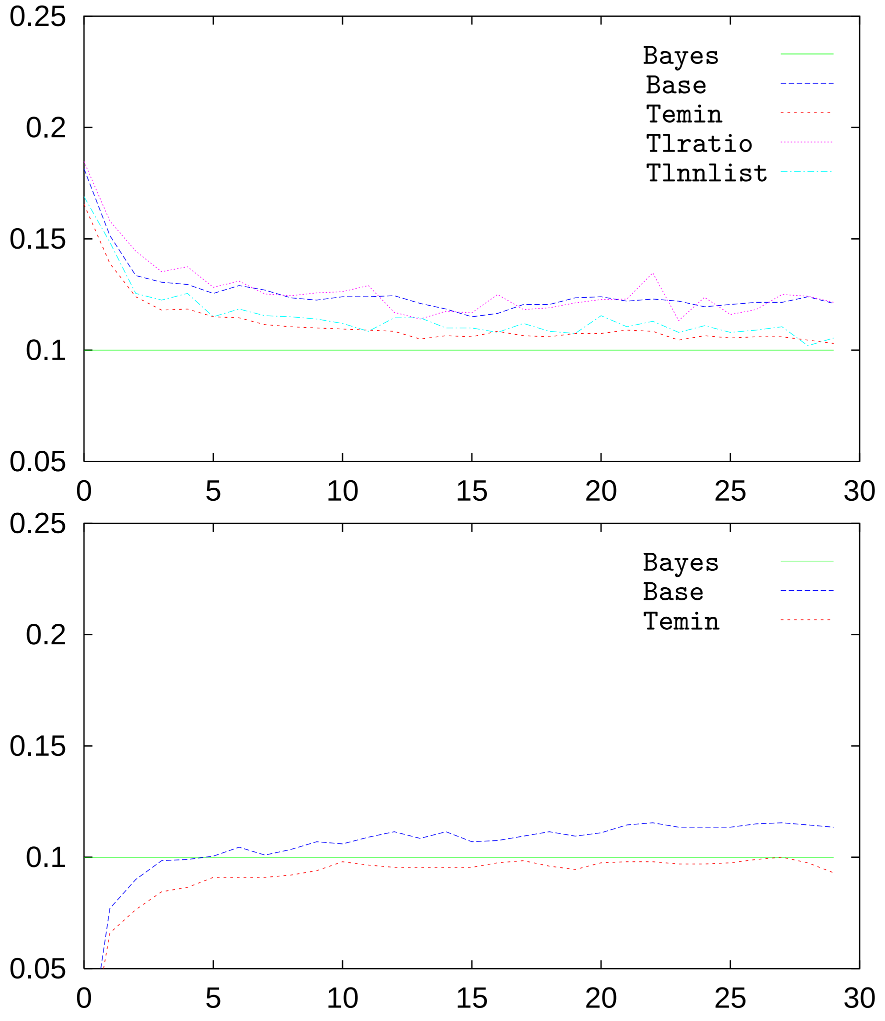

Figure 9: L (above) and R (below) error estimates for $\boldsymbol{I}$-$\boldsymbol{I}$ problem with $n_i = 100$ and $\hat{\boldsymbol{\Sigma}}_i = \boldsymbol{\Sigma}_i$. Results are averaged over 10 independent trials.

results for all other cases, and partly on the fact that we can obtain the exact (erroneous) result in (Fukunaga, 1990) by restricting the weight matrices to $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{I}$.

We make the following observations based on the results in Figures 9-11.

1. Without a threshold selection process, the bias can be quite large. This effect can be seen in the $\boldsymbol{I}$-$4\boldsymbol{I}$ problem where the bias of the `Base` method is extremely large.

2. All threshold selection methods tend to reduce the bias.

3. As expected, selecting the threshold via the **ErrKnnTemin** method (which minimizes empirical error without employing a leave-one-out strategy) can bias the L error estimate low, as seen in the $\boldsymbol{I}$-$\boldsymbol{\Lambda}$ problem.

4. The variance of the error estimates is larger (relative to the error value) for the $\boldsymbol{I}$-$\boldsymbol{\Lambda}$ problem than the other two. As a general rule we expect the variance to be larger for

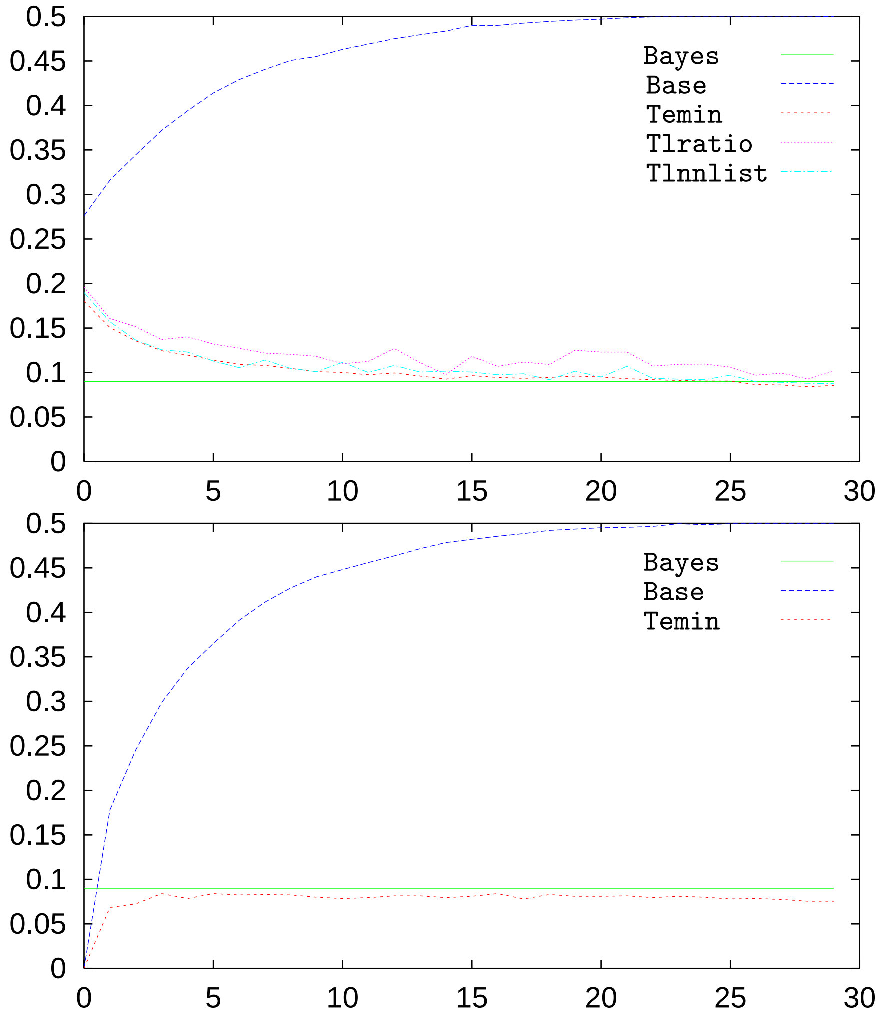

Figure 10: L (above) and R (below) error estimates for $\boldsymbol{I}$-4$\boldsymbol{I}$ problem with $n_i = 100$ and $\hat{\boldsymbol{\Sigma}}_i = \boldsymbol{\Sigma}_i$. Results are averaged over 10 independent trials.

problems with smaller Bayes error. Intuitively this happens because there are fewer samples in error and therefore fewer to contribute to the error estimate.

5. The L and R methods provide correct upper and lower bounds for the Bayes error when used in conjunction with the appropriate threshold selection technique, i.e. the **Temin** technique for the R method and the **Tlratio** or **Tlnnlist** technique for the L method.

6. Overall, the best results are obtained using the **Tlnnlist** technique for the L error estimates and the **Temin** technique for the R error estimates. However, the more efficient **Tlratio** technique for the L estimates also produces good results.

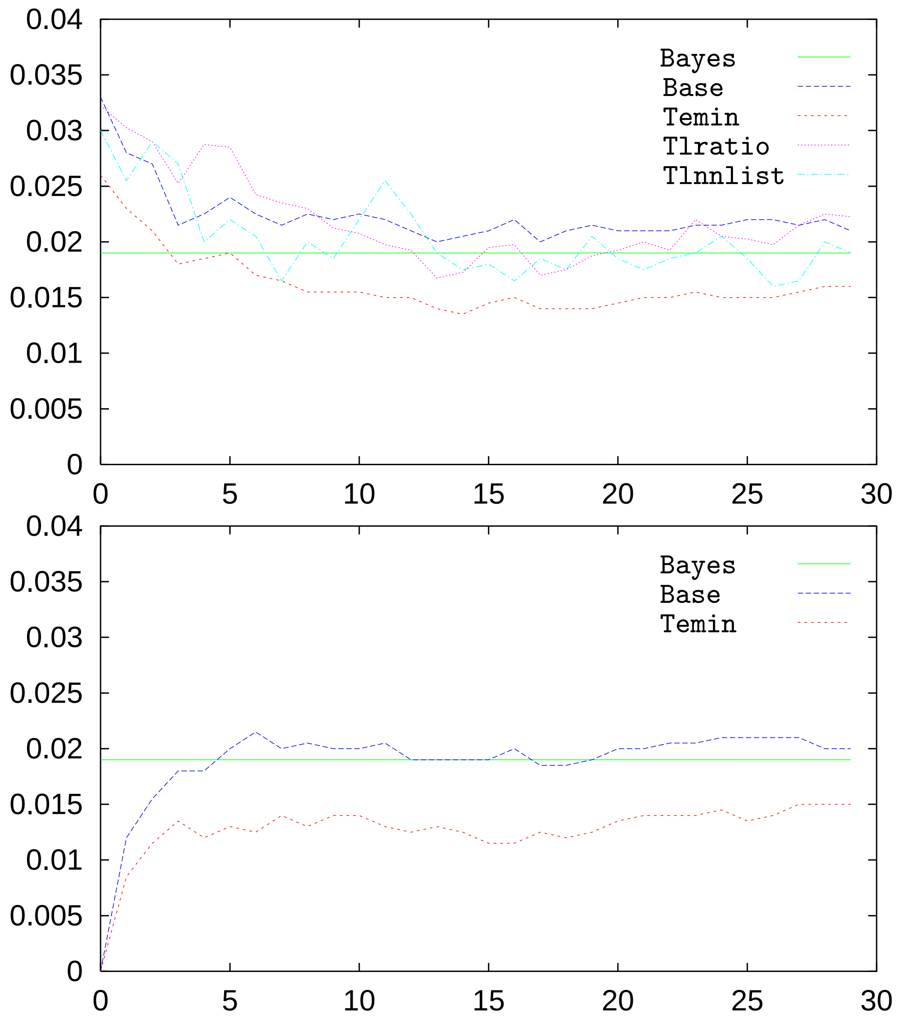

Figure 11: L (above) and R (below) error estimates for $\boldsymbol{I}$-$\boldsymbol{\Lambda}$ problem with $n_i = 100$ and $\hat{\boldsymbol{\Sigma}}_i = \boldsymbol{\Sigma}_i$. Results are averaged over 10 independent trials.

The second set of experimental results is shown in Figures 12-14. These results were obtained in the same way as the others, except that they use a single sample set of size $n_0 = n_1 = 110$, and average the error estimates over 10 bootstrapped sample sets of size $n_0 = n_1 = 100$. In addition, the exact weight matrices are replaced with covariance estimates from the data. These results represent a more realistic view of what one might see in practice. They differ from the previous results in that the variance of all the estimates is higher. This is to be expected, since we are using fewer total samples and are averaging over data sets that are not independent. In addition, both the L and R estimates are biased lower for the $\boldsymbol{I}$-$\boldsymbol{I}$ problem. This bias is due in part to the use of data driven covariance estimates, and in part to the variance associated with the sampling process (i.e. the extent to which this particular set of 110 samples represents the true underlying statistics). Generally speaking, the results presented here are quite good, considering the dimensionality of the problem and the relatively small sample sizes.

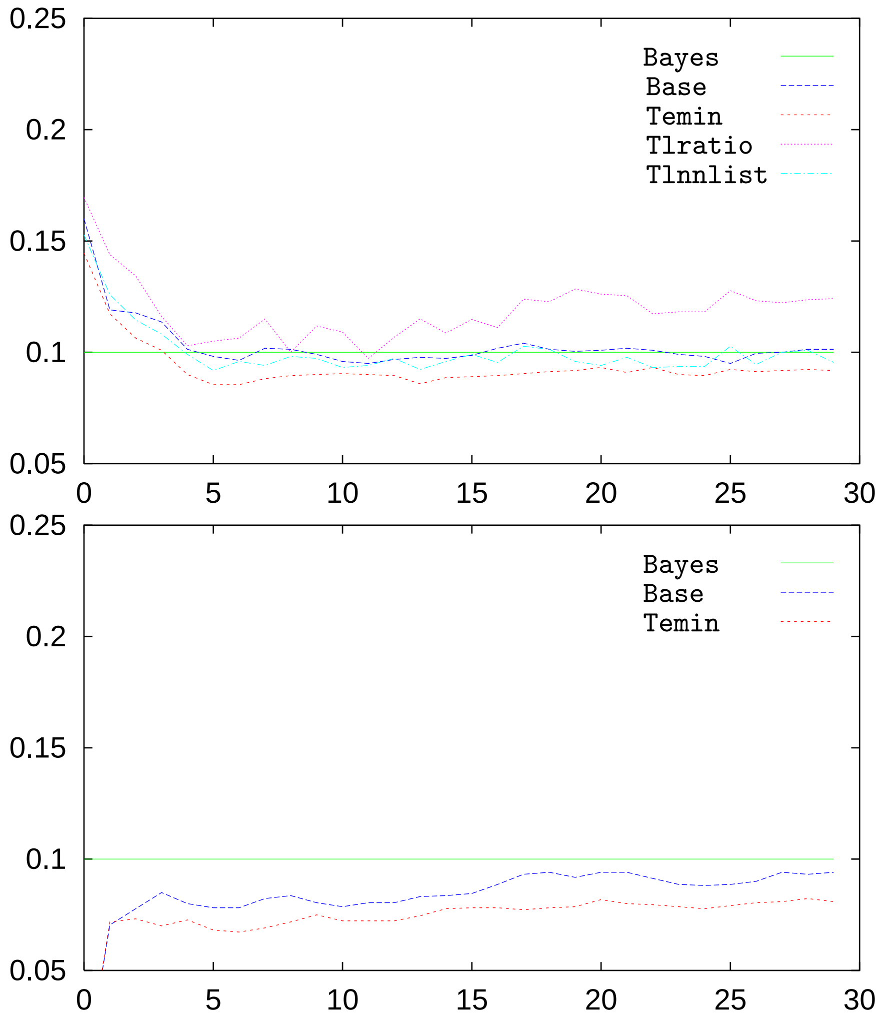

Figure 12: L (above) and R (below) error estimates for ***I-I*** problem with $n_i = 110$ and $\hat{\boldsymbol{\Sigma}}_i$ estimated from the data. Results are averaged over 10 bootstrapped sets of size $n_i = 100$.

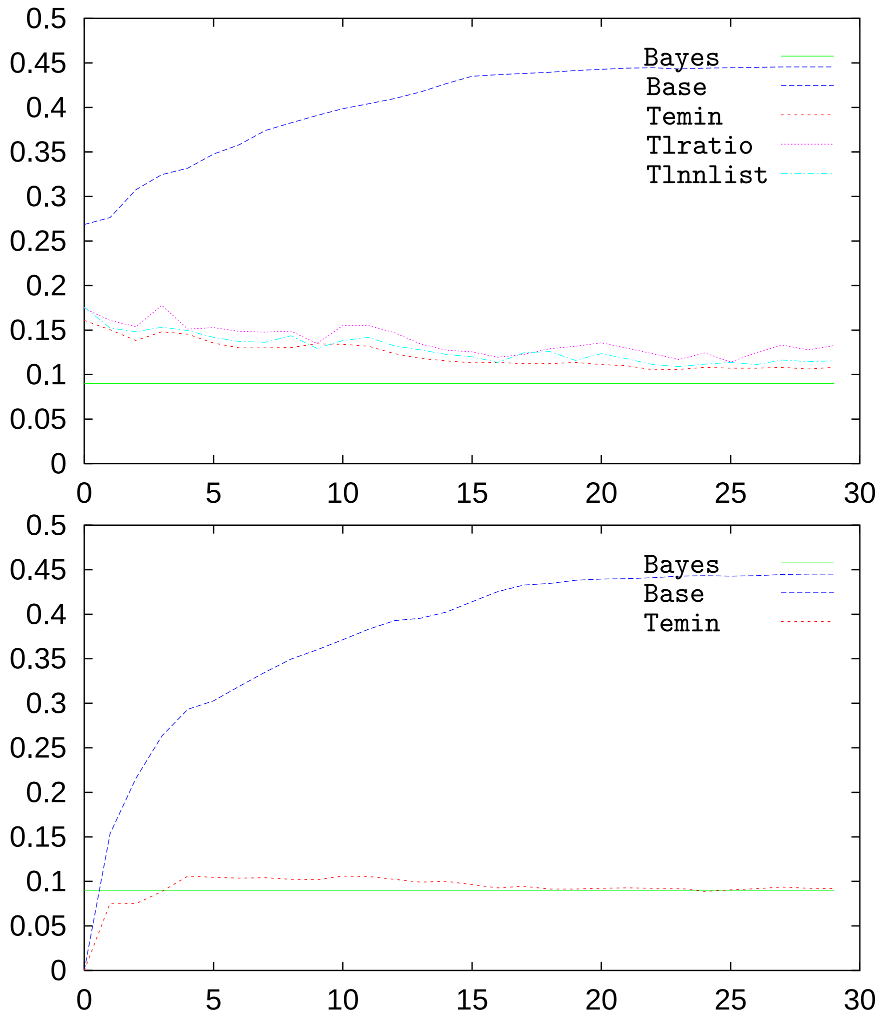

Figure 13: L (above) and R (below) error estimates for $\boldsymbol{I}$-$4\boldsymbol{I}$ problem with $n_i = 110$ and $\hat{\boldsymbol{\Sigma}}_i$ estimated from the data. Results are averaged over 10 bootstrapped sets of size $n_i = 100$.

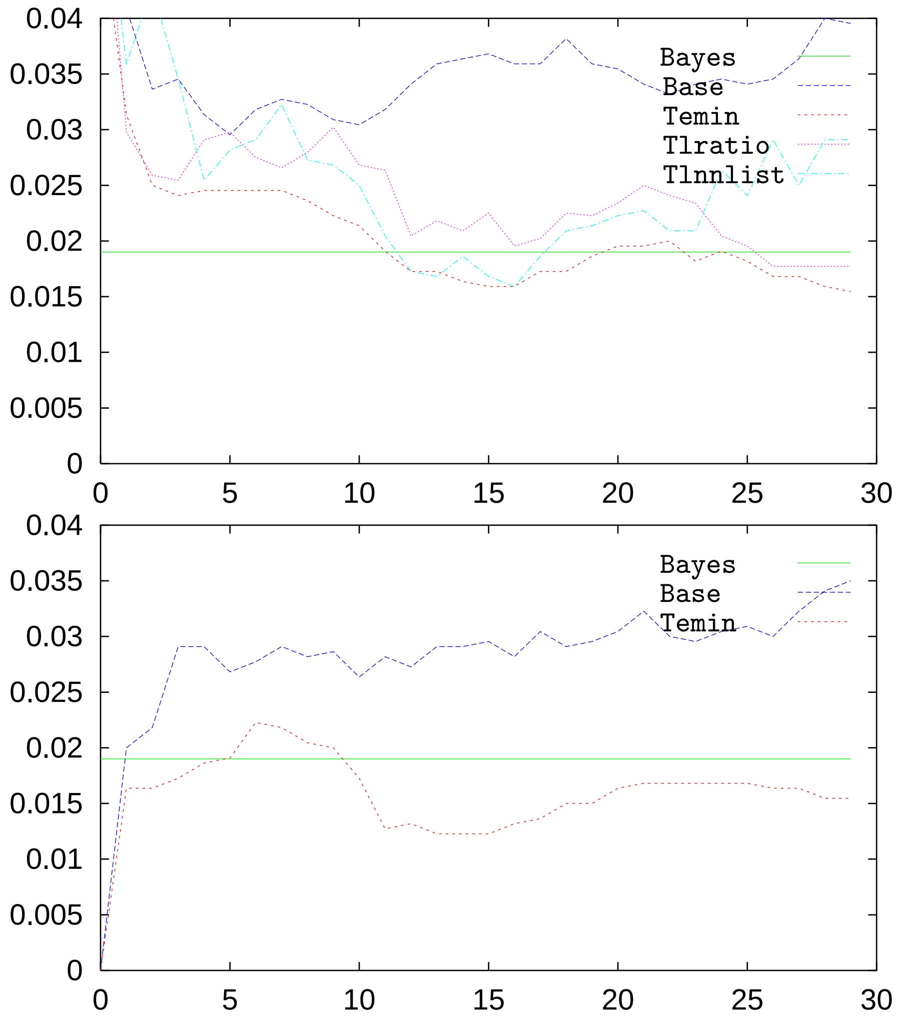

Figure 14: L (above) and R (below) error estimates for $\boldsymbol{I}$-$\boldsymbol{\Lambda}$ problem with $n_i = 110$ and $\hat{\boldsymbol{\Sigma}}_i$ estimated from the data. Results are averaged over 10 bootstrapped sets of size $n_i = 100$.

9 Combining Bias Reduction and Curve Fitting

The following is a straightforward curve-fitting procedure for estimating the Bayes error: use the brute force procedure in Algorithm 1 (which uses $\Delta\tau = 0$) to produce L error estimates for $k = 2, 3, ..., k_{max}$, and then form a Bayes error estimate by fitting the bias expression in Equation (26) to the $\hat{\epsilon}_n$ versus k data ($\hat{\epsilon}^*$ is the constant term from the fit). Unfortunately, this approach tends to produce poor estimates of the Bayes error. Reasons for this are best understood by considering the sources of error associated with this approach.

1. The bias expression in Equations (26) is not exact. It represents a low-order approximation that is valid only when the *kNN* classifier closely approximates the Bayes classifier. Simply put, the bias expression is valid only when the bias is small. Since the bias of the error estimates from Algorithm 1 may be large, it is inappropriate to fit these error estimates to the expression in Equation (26).

2. The variance of the sampling process and the error estimation procedures add uncertainty to the error estimation process. Empirically we have observed the variance of the R error estimates to be much smaller than those of the L error estimates. This observation suggests that a more reliable estimate of the Bayes error may be produced by curve fitting the R error estimates.

3. The variance of the curve-fitting procedure also adds to the uncertainty of the estimation process. The simplest way to reduce this variance is to reduce the number of free parameters determined by this procedure.

To produce reliable estimates of the Bayes error, we must develop strategies that are effective in controlling these sources of error. With this in mind, we propose the following two methods (the second is an extension of the first).

Method 1: This method introduces a new technique for adjusting the threshold that couples the threshold selection process to the curve fitting procedure. The same threshold is used for all data samples and all values of k. Consider the following expression for the bias (a greatly simplified version of (26))

$$\bar{\Delta}\epsilon \cong b_t \Delta t^2 + b_1 \left(\frac{1}{k}\right) + b_2 \left(\frac{k}{\mathrm{n}}\right)^{2/\mathrm{d}} + b_3 \left(\frac{k}{\mathrm{n}}\right)^{4/\mathrm{d}} \tag{30}$$

Note that this expression differs from the one in (27) in the $b_t\Delta t^2$ term. We believe this to be an important term that was neglected in (Fukunaga, 1990). The basic idea is to determine the threshold $\hat{\tau}$ that minimizes the effect of the $(k/N)^{2/\mathrm{d}}$ and $(k/N)^{4/\mathrm{d}}$ terms in the bias expression. We can show that both b_2 and b_3 depend on Δt^2, but b_1 does not. Note also that we can relate t to $\hat{\tau}$ through the formula $\tau = \left(\frac{\mathrm{n}_1|\mathbf{\Sigma}_1|}{\mathrm{n}_0|\mathbf{\Sigma}_0|}e^{-t}\right)^{2/d}$. So we wish to choose the threshold to minimize the coefficients b_3 and b_4, or more simply to minimize the combined contribution of these two terms. An analytical expression for the optimal threshold is both difficult to derive and inevitably a function of unknown quantities. But the desired threshold can be determined empirically as follows. Observe that without the $(k/N)^{2/\mathrm{d}}$ and $(k/N)^{4/\mathrm{d}}$ terms the bias takes on the form

$$\bar{\Delta\epsilon} \cong b_t \Delta t^2 + b_1 \left(\frac{1}{k}\right) \tag{31}$$

which is approximately constant for large k, i.e. $\hat{\epsilon}_n \cong \epsilon^* + b_t \Delta t^2$ for large k. With this observation, our strategy is to seek a threshold $\hat{\tau}$ that produces error estimates that are approximately constant for large k. This strategy can be implemented in a brute force manner by sequentially exploring a range of threshold values, fitting a constant to the large k error estimates, and retaining the threshold that produces the fit with smallest variance. Preliminary results from this approach are quite promising. The constant from the fit provides a good estimate of the Bayes error. Note that it minimizes the contribution of all terms from the bias expression except $b_t \Delta t^2$. It also minimizes the error introduced by the curve-fitting procedure by reducing it to a single parameter estimation technique (i.e. fitting a constant). Finally, this method lends itself to an efficient implementation (beyond the scope of this report) that circumvents the need to run the entire error estimation procedure for each value of $\hat{\tau}$.

Method 2: This method is an extension of Method 1 designed to compensate for the $b_t \Delta t^2$ bias term. It begins by using Method 1 to determine a threshold $\hat{\tau}^*$ and corresponding Bayes error estimate $\hat{\epsilon}^*$. It then adjusts $\hat{\epsilon}^*$ by subtracting an estimate of $b_{t^*}(\Delta t^*)^2$. Note that since t and $\hat{t}^*$ are known, $(\Delta t^*)^2$ is known, so all that is needed is to estimate b_{t^*}. An estimate of b_{t^*} can be obtained by estimating b_1 (the coefficient of the $1/k$ term in the bias expression) and then exploiting a known relationship between b_t and b_1. Specifically we can show that $b_t = 0.5 b_1$. To estimate b_1 we fit the error estimates for small k (under the threshold $\hat{\tau}^*$) to the formula

$$\hat{\epsilon} = b_0 + b_1 \left(\frac{1}{k}\right)$$

Once we have an estimate of b_1, we use the formula below to produce the adjusted estimate of the Bayes error

$$\hat{\epsilon}^*_{adj} = \hat{\epsilon}^* - 0.5 \hat{b}_1 (\Delta t^*)^2$$

References

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*, Vol. 31 of *Applications of Mathematics.* Springer-Verlag, Inc., New York, NY.

Devroye, L., & Wagner, T. (1979). Distribution-Free Inequalities for the Deleted and Holdout Error Estimates. *IEEE Transactions on Information Theory*, *25*(2), 202–207.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc., New York, NY.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition* (2nd edition). Academic Press, Inc., San Diego, CA.

Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*, Vol. 26 of *Statistics and Applied Probability*. Chapman & Hall, London, UK.

Tou, J., & Gonzalez, R. (1974). *Pattern Recognition Principles*, Vol. 7 of *Applied Mathematics and Computation*. Addison-Wesley Publishing Co., Inc., Reading, MA.